PROACTIVE CYBERFRAUD DETECTION THROUGH INFRASTRUCTURE ANALYSIS

Andrew J. Kalafut

Submitted to the faculty of the Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in Computer Science

Indiana University

July 2010

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Doctoral Committee

Minaxi Gupta, Ph.D.
(Principal Advisor)

Steven Myers, Ph.D.

Randall Bramley, Ph.D.

July 19, 2010

Raquel Hill, Ph.D.

To my family

Acknowledgements

I would first like to thank my advisor, Minaxi Gupta. Minaxi's feedback on my research and writing have invariably resulted in improvements. Minaxi has always been supportive, encouraged me to do the best I possibly could, and has provided me many valuable opportunities to gain experience in areas of academic life beyond simply doing research.

I would also like to thank the rest of my committee members, Raquel Hill, Steve Myers, and Randall Bramley, for their comments and advice on my research and writing, especially during my dissertation proposal.

Much of the work in this dissertation could not have been done without the help of Rob Henderson and the rest of the systems staff. Rob has provided valuable data, and assisted in several other ways which have ensured my experiments have run as smoothly as possible.

Several members of the departmental staff have been very helpful in many ways. Specifically, I would like to thank Debbie Canada, Sherry Kay, Ann Oxby, and Lucy Battersby.

I would like to thank my co-authors on all the work that has contributed to this dissertation. Material in Chapter 2 [42] is based on work co-authored with Minaxi Gupta. Material in Chapter 4 [58] is based on work co-authored with D. Kevin McGrath and Minaxi Gupta. Material in Chapter 5 [41] is based on work co-authored with Minaxi Gupta, Pairoj Rattadilok and Pragneshkumar Patel. Material in Chapter 6 [40] is based on work co-authored with Minaxi Gupta, Christopher Cole, Lei Chen, and Nathan Myers. Material in Chapter 7 [43] and Chapter 8 [93] is based on work co-authored with Craig Shue and Minaxi Gupta.

My parents, my sister, and my brother have been a great source of support throughout my life, and encouragement in pursuing my education. I am greatful for their support.

I would especially like to thank my financée, Heather Goodman, for her love and support.

Finally I would like to thank my friends who have helped make life in graduate school more fun and less stressful than it otherwise would have been, especially Craig Shue, Joshua Hursey, Samantha Foley, Alia Al-Kasimi, Eric Nichols, and Abhinav Acharya.

Andrew J. Kalafut

PROACTIVE CYBERFRAUD DETECTION THROUGH INFRASTRUCTURE ANALYSIS

Internet users are threatened daily by spam, phishing, and malware. These attacks are often launched using armies of compromised machines, complicating identification of the miscreants behind the attacks. Unfortunately, most current approaches to fight these problems are reactive in nature, allowing significant damage before security measures are adapted to new attacks. For example, blacklisting prevents communications with known malicious hosts, but many users may fall victim to an attack before blacklists are updated. In this dissertation we argue for a proactive approach to fighting cybercrime. Our approach relies on the observation that to avoid attribution and to stay up amidst take-down attempts, miscreants must provision their infrastructure differently than legitimate web sites. Thus, we propose to proactively identify malicious activity using unique characteristics of malicious web site provisioning. Specifically, using near real-time feeds of malicious web hosts, we investigate the extent to which miscreants use five specific provisioning practices. The first three are based on the Domain Name System (DNS), which translates host names to IP addresses. We first examine fast-flux, a practice where the association between name and address changes much more frequently than usual. We then investigate the use of DNS wildcards, which point many host names to a single address. Next, we examine the use of orphan DNS servers, which are DNS servers in non-existent domains. Then, we study the concentration of malicious activity in certain networks. Finally, we examine web redirects, which may appear to be links to legitimate web sites but in reality trick users into visiting malicious sites. We find that although good web sites sometimes make use of some of these techniques, malicious web sites are more likely to use them. Consequently, their presence can be used for proactive identification of malicious web sites.

Contents

List of Tables

List of Figures

1

Introduction

Internet users are threatened by *scam*, *phishing*, and *malware* on a daily basis. Although many of these cyberfraud attacks begin through channels such as email, Web searches, and online social networks, they often lead users to web sites controlled by miscreants. There, unsuspecting users may be victimized by scams or tricked into revealing sensitive personal information. Further, their computers may be infected with malware, potentially becoming *bots* under the control of a miscreant, from which further attacks may be launched.

These attacks are lucrative for the miscreants, and the risks to users from them is on the rise. According to an Anti-Phishing Working Group (APWG) report [1], phishing attacks more than doubled from the first half of 2009 to the second half. According to the Internet Crime Complaint Center (IC3) 2009 Internet Crime Report [35], 336,655 complaints were submitted in 2009, a 22% increase over 2008. The total dollar losses from cases they referred to law enforcement was $559.7 million, the highest since they began operation, and over double the 2008 amount. Also, a January 2010 Consumer Reports survey [12] concludes that in the previous two years, American consumers lost $4.5 billion to viruses, spyware, and phishing.

1.1 Current Approaches

Many current approaches exist to protect users from the problems we address. The major approaches currently in use include blacklists and anti-virus software. Blacklists are lists of malicious Uniform

Resource Locators (URLs), domains, or IP addresses, which users should be prevented from visiting. Blacklists are often used by browser toolbars, search engines, and spam filters, to warn users or prevent them from visiting malicious web sites. Anti-virus software typically runs on end user computers and looks for binary files containing certain byte sequences matching known malware. This detection requires the malware to be downloaded before it can be identified. More advanced anti-virus looks for specific sequences of system calls used by known malware. This detection comes later, after the malware has started running.

Given how commonly users are reported to fall prey to these attacks, it is clear that the approaches described above are not effective enough. In part this is attributable to a common drawback they all share: they are *reactive*. As a result, they may allow significant damage before they are adapted to new attacks. Consider blacklisting as an example. A malicious web site has to be discovered, often through slow means such as user reporting, verified, added to a blacklist, and new copies of the blacklist distributed, before the web site is blocked. Since the same malicious URL may be emailed to a large number of recipients in a phishing attack, there is large opportunity for many victims to visit the malicious site before any realize it is malicious and report it. The problem is essentially the same for anti-virus software as well. There is an inherent lag between the time a new attack appears and the time a signature is generated for it.

The reactive nature of these approaches has another unfortunate side effect: many attacks may be missed. Consequently, some attacks may only ever get reported to maintainers of one blacklist or anti-virus product and not others. This leads to a difference in what can be detected depending on the product being used. In fact, we observe large differences in blacklists meant to serve similar purposes as each other. In two feeds of phishing sites we receive, one contained 36,067 phishing URLs in June 2010, while another only contained 14,879 URLs during the same time period. Each list misses many URLs; Only 7,174 were in both lists.

Actual take-down of a malicious web site avoids the issues in blacklist coverage, protecting all users regardless of which security products or blacklists they may be using. However, for many reasons including communications difficulties and the indifference of some Internet Service Providers (ISPs) to deal with the problem, this also comes too late, after malicious web sites may have already done some damage. This is corroborated by the fact that the median life time for a scam site is over

a week [4]. Phishing site life times are shorter, but still average 31 hours after being detected [1].

By dealing with only certain domain names or IP addresses currently involved in attacks, or certain types of malware, current methods only deal with the symptoms of attacks. Because they do not focus on the behavior of the attackers, these approaches do nothing to solve the overall problem. Consequently, when a malicious web sit is shut down, attackers are free to continue similar slightly modified attacks. Therefore, an approach that identifies common features of attacks instead of individual attacks themselves is likely to be useful.

1.2 Our Approach

In this dissertation, we work towards the goal of *proactive* identification of malicious web sites involved in cyberfraud such as phishing, malware, and scams. We take an experimental approach to the problem, examining malicious web sites to develop light-weight *features* that can be used for proactive identification. Our features are motivated by the observation that attackers have differing requirements from benign web site operators for provisioning the infrastructure for their web sites. When a good web site operator sets up a web server, they are mainly concerned with providing high quality service to their users. They may over-provision to deal with occasional attacks. In contrast, malicious web site operators must provision for availability in spite of take-down attempts by law enforcement or their ISP. An additional unique concern of attackers is escaping detection and attribution. Due to this concern and others, attackers often provision their web sites using *botnets*, armies of compromised machines. This practice exacerbates their availability problems, since these compromised machines may be cleaned or shut down at any time. A further concern of malicious web site operators is protecting the most valuable parts of their infrastructure, such as faster computers, computers outside of firewalls, or those with public IP addresses. Another concern of malicious operators is the desire to be resilient to blacklisting of IP addresses and host names they are using. We hypothesize that to accomplish these goals, miscreants must provision the infrastructure of their web sites differently from benign web sites.

Our approach to proactive identification of malicious web sites is to identify these sites using the unique features of their infrastructure provisioning. We use the term infrastructure provisioning

to include all aspects of how these web sites are provisioned. This includes how the malicious web site operators set up their web and Domain Name System (DNS) servers, how they connect these sites to the Internet, and how the operators attract visitors to their sites. Using knowledge about the provisioning practices beneficial to attackers, we aim to develop features which can be used to proactively identify malicious web sites, preferably before they attract visitors. Because of the large number of domains in the Internet, currently over 193 million [107], proactive identification would be impossible if the features were slow. Therefore, we additionally require our features to be light-weight.

Web sites that are determined to be malicious based on their infrastructure provisioning should then be further scrutinized or blocked. We envision two ways this could be realized. One method would be a monitoring system. Such a system would harvest URLs and domain names from several sources including spam trap email addresses and traffic seen by routers cooperating with the monitoring system. The monitoring system would apply tests for infrastructure features to each site, and using a classifier, flag potentially malicious sites. This information could then be published in a blacklist, different from current blacklists by the scope of coverage and speed in identifying malicious web sites. A second possible method of applying our features would be by programs at end hosts, most likely web browsers. They would test each URL they visit, especially if the link to the URL came from an external source such as an email program or a domain other than the one the link is leading to.

Specifically, our contribution in this dissertation is the investigation of five features of malicious web site infrastructure provisioning that may be used in a framework to detect malicious web sites. We examine fast flux, the use of DNS wildcards, the use of orphan DNS servers, disproportionately malicious Autonomous Systems (ASes), and use of redirects. We focus on showing that each feature can identify a subset of malicious activity and is light-weight, not on the effectiveness of any particular combination of features.

1.3 Organization

We organize the rest of this dissertation as follows.

1.3.1 Background

Before examining each feature of malicious web site infrastructure provisioning, some background knowledge is necessary. We discuss three aspects of background material. First, because the first three of the infrastructure properties we examine relate to the domain name system, in Chapter 2 we discuss DNS, including a measurement study of which types of DNS records are used most. Next, because the work in this dissertation examining each infrastructure property is highly data-driven, we describe our data sources in Chapter 3. Finally, after examining our infrastructure features, in Chapter 9 we discuss related work, including work related to each feature, as well as other malicious infrastructure, and complementary approaches.

1.3.2 Infrastructure Features

We investigate five features of infrastructure provisioning which may be useful in a framework for identification malicious web sites. We devote a chapter to exploring each one.

Fast Flux

The first infrastructure provisioning practice we investigate, in Chapter 4, is *fast flux* [58]. Fast flux is a technique used by miscreants to provide high availability of their web sites in spite of take-down attempts. Take-down attempts can focus in two areas, revoking the domain name used by the miscreants, and cleaning the bots the miscreant is hosting the attack on. Fast flux helps miscreants mitigate the effects of the latter. The host name of a web site taking advantage of fast flux resolves to a large number of IP addresses. These addresses are geographically diverse and change often. Using this technique the malicious web site is little affected by the removal of a single host. We find fast flux in use by 11% of phishing web sites. Since there is no known reason for a legitimate web site to use this technique, fast flux is a good feature for identifying malicious web sites. However, since not all use it, other features are needed as well.

DNS Wildcards

In Chapter 5, we examine *DNS wildcards* [41]. A DNS wildcard can be used to point arbitrary requests for host names within a domain to a specific host name or IP address. Wildcards offer administrators the convenience of not having to change DNS entries when host names change. Wildcards are one of the original features of DNS, having been defined in the original standard. As such, they have legitimate uses. However, they are also attractive to miscreants. Specifically, wildcards help them evade blacklists by allowing them to use hundreds of host names without changing DNS entries, thereby reducing the effect blacklisting a single host name has on their activities. Although it is possible for miscreants to use many host names in a domain without wildcards, the overhead would be higher, as they would need DNS entries for each. In this work, we investigate malicious and legitimate uses of wildcards. We find wildcards to be in more use by malicious sites than by good sites, with 75% of scam sites using them, as compared to 25% of good sites. In the cases where good sites use wildcards, we find other features, such as the Time to Live (TTL) associated with the wildcard record, that can help us distinguish good and malicious wildcard uses. Although it may not be useful for identifying malicious web sites on its own, DNS wildcards appear to be a good feature for identifying malicious web sites when used in conjunction with other features.

Orphan DNS Servers

An *orphan DNS server* [40] is a DNS server which has an address record in the DNS system, even though the domain in which it resides has no DNS records itself and hence does not exist. For example, the DNS server `ns.foo.com` would be considered an orphan DNS server if `foo.com` is a non-existent domain. Orphan name servers may arise from domains being deleted, without the address records for name servers within the domain also being deleted. These name servers may be left to account for other domains which may be using them. However, if the removal of the domain was part of a take-down, then the other domains using these name servers are likely to be malicious as well. In Chapter 6 we examine the uses of orphan DNS servers by both malicious and good sites. We find 46.8% of domains using orphans indirectly associated with malicious activity. Because this

association is indirect, and applies to fewer than half of domains using orphans, on their own orphans are not a good indicator of malicious activity. However, we believe that due to the possibility of them arising out of malicious activity, use of orphan DNS servers may be an effective feature when combined with others.

Malicious Autonomous Systems

Malicious activity is not necessarily evenly distributed across the Internet: some networks may employ lax security, resulting in large populations of compromised machines, while others may tightly secure their network and do not have any malicious activity. In Chapter 7, we search for *disproportionately malicious ASes* [43], the organizational units that comprise the Internet. In doing so, we make an effort not to penalize large ASes for having more malicious activity simply due to being large. We find certain ASes with more malicious activity than normal for their size. Although legitimate web sites may be hosted in such ASes, association with such ASes should be regarded as suspicious. Thus, we believe that the use of hosts in ASes with disproportionate malicious activity can be used as a feature to identify web sites likely involved in malicious activity.

Web Redirects

The final provisioning feature we investigate, in Chapter 8, is the use of web redirects. A redirect is a URL which immediately instructs the web browser to go to a different page. It has several uses, including tracking the which links on a web site users click on. Redirects may be useful to miscreants since they may appear to users as links to a web site the user trusts, but actually lead them to a different web site under control of the miscreant. This investigation is in two parts, first we investigate *open redirects* [93], and then redirects leading to malicious web sites. An open redirect is one which may be manipulated to point to any destination. Such a redirect may be useful to an attacker if the site the link appears to load to is a site users may trust. They can then be tricked into clicking such a link which appears as if it would lead to a trusted page, but instead leads to a page chosen by the miscreant. This can be especially dangerous of the miscreant designs their page as a look-alike of the page the user thinks the redirect will lead to. We find that a large percentage

of links identified as redirects by our heuristics are open, and thus can be exploited by miscreants. We then investigate actual use of redirects to lead to malicious web sites, and find that over three quarters of URLs in a phishing blacklist are redirects, much more than are found in a directory of legitimate sites. This extensive use of redirects leading to malicious web sites indicates that they would be a useful feature for identification of such sites.

1.3.3 Conclusion

We conclude the dissertation in Chapter 10 with a short summary of our findings. Although good web sites are sometimes making use of some of the provisioning practices we explore, we find that malicious sites are more likely to use them. Each of these light-weight features can be used to identify a subset of malicious web sites. Additionally, each may incur false positives. Therefore, it is likely to be beneficial to use them together, perhaps as features in a classifier. We plan to pursue this direction in future work.

The investigation in this dissertation has led us to believe that a system based on our framework would be a good first line of defense for combating web-based cyberfraud. However, crime on the Internet is sophisticated and diverse. Given the variety of approaches taken by miscreants, no single strategy alone is likely to be a perfect defense against all cyberfraud. For this reason, we envision that our approach would be complementary to current anti-cyberfraud approaches, not a replacement.

Finally, we realize that the techniques miscreants use to evade detection and to stay alive in the face of take-down attempts are likely to change over time. As they change strategy, some of our features may become obsolete. However, evading all of the features described here would be difficult in practice and take miscreant considerable time. Additionally, the spirit of our features is fundamental to cyberfraud. Miscreants will always face the same underlying constraints: the desire to be untraceable, and the need to face take-down attempts. As the strategies used by miscreants to deal with these constraints evolve, the features in our framework also may be periodically refined, with new ones added to deal with new provisioning strategies.

2

Domain Name System

2.1 Introduction

The Domain Name System (DNS) [63,64] is an important component of the Internet's infrastructure. The primary purpose of DNS is to translate host names to IP addresses. Internet users rely on the DNS every time they wish to address a host by its name. Three out of the five infrastructure properties examined in this dissertation, DNS wildcards, orphan DNS servers, and fast flux, are related to DNS. To understand these chapters better, in this chapter we present an overview of the DNS.

2.2 DNS Structure

The DNS is a hierarchical system, organized as a tree. The entire DNS space is divided into various *zones*. Each zone consists of a connected portion of this tree under the same administrative control. This tree begins with the *root* zone, whose name consists of just a single dot (`.`). Directly above the root are Top Level Domains (TLDs), zones such as `.com` or `.de`. Above these are zones for second level domains, also referred to as *domain names* such as `example.com` or `indiana.edu`. This may continue for several more levels.

In order to operate properly, each DNS zone must have at least one name server[1] which serves

[1]The terms *DNS server* and *name server* are used interchangeably.

the DNS records within that zone. Normally, there is more than one name server for a zone, with one being designated as the *primary name server* and any others being designated as *secondary name servers*. The primary server for a zone will load all the DNS records for the zone from a *zone file*. Secondary servers will get this information through the *zone transfer* operation, which transmits all of the DNS records in the zone, or they will load the same information from the zone file themselves.

The DNS servers at each zone store and respond to queries about hosts within their zone. They also contain `NS` records pointing to the DNS servers for their *subzones*. Each dot in a host name usually indicates a host in a zone or a subzone. For example, `www.indiana.edu` is a host within the `indiana.edu` zone, while `cs.indiana.edu` is a subzone of `indiana.edu`. The `indiana.edu` name servers will have records indicating the names of the `cs.indiana.edu` name servers.

2.3 DNS Records

Information is stored in the DNS in the form of `records`. DNS records consist of five basic parts, numbered in Figure 2.1.

www.iub.edu.	50400	IN	A	129.79.78.186
1	2	3	4	5

Figure 2.1: Sample DNS record

1. The first component of a DNS record is the *name*. This is often a host name. The name (along with the type and class) are used to look up the appropriate DNS record in a query. In the example in Figure 2.1, the record would be returned for queries for in `www.iub.edu`. Such a query would typically be made to get the address of the server when a user tries to visit this web site.

2. The next component shown in this sample DNS record is the *Time to Live (TTL)*. After receiving the result of an DNS query, the local DNS server, and often the local computer, cache the result to lower overhead for future queries for the same record. The TTL is the number of seconds the record may be used from cache.

3. The *class* field is little used. Although a few other values have been used in the past for various purposes, all DNS records on the Internet are in the `IN` class.

4. The *type* field specifies what type of data the record contains. In this example, the type `A` indicates that this record contains an IPv4 address. Details of other popular types will be discussed in Section 2.5.

5. Finally, every DNS record contains some data. The structure of the data depends on the type of DNS record. In this case it is an IP address, but can also contain another host name or other information depending on the record type.

2.4 DNS Query Resolution

Each organization has two types of DNS servers. The first, internal DNS servers, typically referred to as *resolvers*, are only for use by clients internal to the organization. The resolvers query remote DNS servers on behalf of the clients and maintain a cache of the received responses. The second, external DNS servers, typically referred to as *authoritative DNS servers*, only respond to queries from remote resolvers for hosts belonging to the organization. Each zone in the tree must have a set of authoritative DNS servers which answer DNS requests about the zone, and give the location of authoritative DNS servers of its subzones. To prevent external clients from discovering the what DNS queries are made by internal clients, and to prevent external clients from discovering internal network structure, it is recommended to separate the resolver and authoritative DNS servers. Sometimes both servers are on the same physical machine, but logically separated.

When a client needs a host name resolved, it contacts its local DNS resolver, whose address it is configured with. When a resolver receives a client query to resolve a host name to an IP address, it follows a series of steps to respond to the request. First, it consults its local cache. If it finds an appropriate valid record for that information, it returns that record to the client. Such a record may be in the cache if another client contacted the resolver earlier for the same record. Otherwise, the resolver traverses the DNS hierarchy to satisfy the client request.

As an example, consider the case when the resolver has to consult the DNS hierarchy to find the

IP address (`A` record) for `www.iub.edu`, because no records for this host name are in the cache. The resolver contacts one of the root DNS servers, requesting the `A` record for `www.iub.edu`. It is able to contact a root DNS server, because the addresses of these are static information, configured at the resolver. The root DNS server does not know the address for `www.iub.edu`, but it does return the address of the DNS servers for `.edu`. The resolver then queries the `.edu` DNS servers, once again asking for the `A` records for `www.iub.edu`. It also does not know the answer, but returns the DNS servers for `iub.edu`. Finally, these DNS servers are queried and the answer is returned. If some intermediate information were cached, not all of these steps would be necessary. If the resolver has cached the DNS servers for `iub.edu`, it can query these directly. If not, it is still likely to have the DNS servers for `.edu` cached, saving the step of querying the root servers.

2.5 DNS Contents

In order to research malicious uses of DNS provisioning, we first undertook a study to understand DNS contents in depth. We describe the study briefly here.

2.5.1 Data Collection

We use two data sets in this section. The first, `zone_transfer`, was obtained by attempting to transfer the zones listed in the `.com` and `.net` TLD zone files [109]. There were 65,101,733 second-level domains in the `.com` zone file and 9,224,482 under `.net` zone file in June 2007, when we started the data collection. Combined, these 74,326,215 domains represented about 58% of the 128 million zones registered at the time [105].

The TLD zone files provide us a list of name servers for each of the second level domains they contain. We attempted a zone transfer from each name server for each zone until we either successfully transferred the zone, or the zone transfer failed for all its name servers. Additionally, if two zone transfers from the same IP address failed, or upon request from the DNS server's administrator, we discontinued making further attempts to transfer any zone from that IP address. Upon connection establishment failure, we retried once. This process took a total of three months. We succeeded in

transferring zones for 4,947,993 (6.6%). We note that in popular DNS software, it is easy to disallow zone transfer altogether, or allow transfer to only a specific set of IP addresses [37]. Therefore, anyone who wishes to make their zone information private can easily make the information unavailable to unauthorized clients. This is the likely cause of a majority of the failures.

One might argue that the `zone_transfer` data set represents zones that are less security conscious because they allow a zone transfer in the first place. To compensate for this limitation, we collect a second data set, `dnssec`. This data set is from security-conscious zones that deploy DNS Security Extensions (DNSSEC) [6]. DNSSEC is a set of extensions to the DNS which provide origin authentication and integrity to DNS data, and authenticated denial of existence. As a side effect, it allows an enumeration of all host names in the zone. This process is slow but allows retrieval of all the records in a zone, just like a zone transfer does. To build this data set, we began with a list of 862 zones with DNSSEC in production usage from the SecSpider DNSSEC Monitoring Project [78]. We limited this to the second level zones within the `.com` and `.net` TLDs to allow a fair comparison with the zones we transferred data from in the same TLDs. This yielded a total of 124 zones. Surprisingly, we also found 161 zones deploying DNSSEC in our zone transfer data. Since 96 of the zones listed under SecSpider already existed in our zone transfer data, we only had to obtain data from the rest of the 28 zones that did not allow a zone transfer.

We used the DNSSEC Walker tool [39] to collect this data. This tool relies on the presence of `NSEC` or `NXT` records which should be present in zones deploying DNSSEC. These records provide a way to discover all of the records from within a zone without using zone transfer. Of the 28 zones we attempted to walk, 4 were only partially walkable due to missing some `NSEC` or `NXT` records. The remaining 24 were completely walkable allowing us to get the same information as we would through zone transfer without actually using the zone transfer query. Our final `dnssec` data set consists of 189 zones. The size of this data set is limited by the low deployment of DNSSEC.

2.5.2 Record Types

We now examine which DNS record types are most widely used among those domains from which we were successful in obtaining a zone transfer. This helps us determine where to focus efforts in

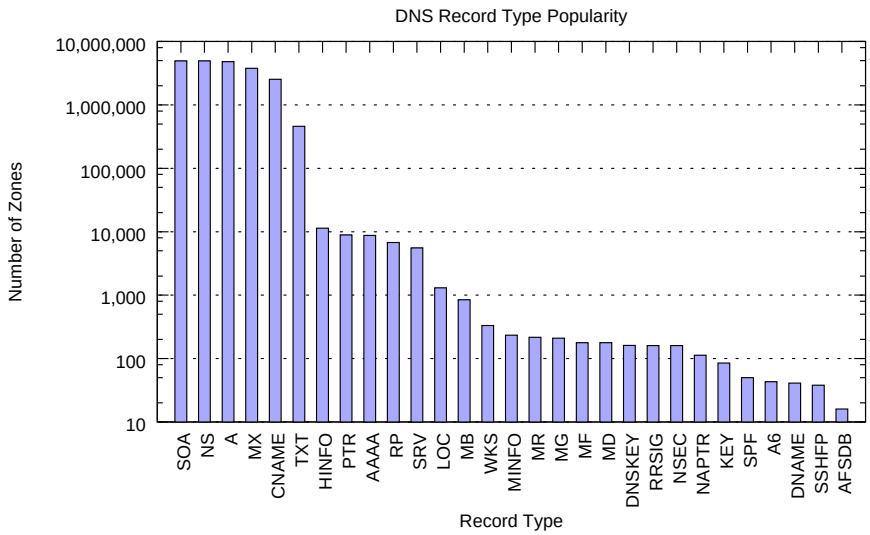


Figure 2.2: Number of DNS zones containing popular record types (log scale)

later DNS related studies.

We see DNS records of 42 types, including the invalid, obsolete, and experimental ones. The most common are described in Table 2.1. Some, such as `SOA`, which contains basic information about a zone, and `NS`, which provides the name of the DNS servers for the domain and its subdomains, are seen in nearly every zone we examine. Interestingly, the `SOA` record, the only record type absolutely required for a zone to exist, is the only one that we see in every zone. Even the vital `NS` is not present in 0.2% of zones, even though it is required by the DNS specification, and despite the fact that we know every one of these zones has at least one name server: the one we used to obtain the zone transfer. Figure 2.2 depicts the number of zones containing each record type seen in 10 zones or more.

From the figure we can see that the top five record types, `SOA`, `NS`, `A`, `MX`, and `CNAME` are much more popular than the others. Material in Chapters 4, 5 and 6 uses the `NS` and `A` record type, with Chapter 5 also using `MX` and `CNAME`. Although `SOA` is present in the most zones we do not consider it, since it is not used for any known malicious purposes.

The next most popular record type after `SOA` is `NS`. This record is used to point to the DNS servers for a domain as well as its subdomains. After `NS`, the next most popular record type is `A`,

Type	Description
SOA	Indicates start of zone. Contact email, primary name server, default TTL, default time zone is valid on secondary name server
NS	Host name of name server
A	IPv4 address
MX	Host name, priority of email server
CNAME	Domain name alias (single name only)
TXT	Arbitrary text
HINFO	Machine and operating system types
PTR	Pointer. Most often used for reverse DNS
AAAA	IPv6 address
RP	Responsible person email and contact info
SRV	Service discovery
LOC	Location. Latitude, longitude, altitude, and precision
MB	Host which contains mailbox for this domain name
WKS	Services available at this domain name
MINFO	Mailbox that should receive error messages for mailing lists at this domain name
MR	Mailbox rename, used for forwarding
MG	Mail group domain name
MF	Host that will forward mail for domain (obsolete)
MD	Host that will deliver mail for domain (obsolete)
DNSKEY	Public key used in DNSSEC
RRSIG	DNSSEC signature
NSEC	Next domain name in zone and record types at that domain name
NAPTR	Rules for URI rewriting
KEY	Public key for DNS transaction signatures
SPF	Sender Policy Framework (email sender authentication)
A6	Experimental IPv6 address record type
DNAME	Alias for domain name plus sub-domains
SSHFP	SSH public key fingerprint
AFSDB	Andrew File System (AFS) database location

Table 2.1: Description of DNS record type contained in 10 zones or more, ordered by popularity.

used to associate a host name with an IP address. The next most popular after these are `MX` and `CNAME`. `MX` specifies the incoming mail server for a domain. `CNAME` aliases one host name to another.

2.5.3 Name Server Redundancy

A legitimate site should want its name servers to be redundant to preserve access to important systems such as their web server if one of their DNS servers should go down. To mitigate the effect of take-down attempts, miscreants may have more redundancy, as we will discuss in Chapter 4.

Here, we briefly examine name server redundancy, to get a better idea of how much redundancy is normally provisioned.

Every zone is required to have at least two name servers [63] and recommended to have at least three [17]. This ensures availability of records when attacks or outages occur. 1,665 zones (0.03%) in the `zone_transfer` data list no name servers at all even though they are required to. Note, however, that this does not make them inaccessible. Clearly, they are accessible since we transferred their zone. Instead, it implies that their `NS` server records existed in their parent zone, but not in the zone itself. This problem does not occur in the `dnssec` data. We find that it is most common to have the required two name servers and no more. In the `zone_transfer` data, 11.9% of zones list fewer than the required two name servers, while 22.1% list more. The `dnssec` zones are provisioned much better, containing only 3% of the zones with fewer than the required two name servers and 66% with more.

2.6 Discussion

While zone transfers yield valuable information for research purposes, the technique raises practical, ethical, and legal questions. We encountered various reactions to our data collection efforts from the zone administrators. Many of the early requests we received were concerns that a machine had been compromised or that we were otherwise attacking their systems. However, over half of the administrators that contacted us were supportive of the work, with a few being being quite enthusiastic. A small number of them requested to have their servers exempted from the scanning, which we promptly honored. The issue of zone transfers has reached the legal system. In a civil court ruling which occurred after our data collection, a North Dakota civil court decision declared unauthorized zone transfers in that state illegal [99]. While the circumstances in that case were unique, it is clear that such queries can be viewed as controversial. Since IP address geo-location software is not always accurate, determining if any servers we transfer from were in North Dakota, or any other jurisdiction where zone transfers may be disapproved of by the legal system, would be very difficult. For these reasons, we choose not to take advantage of zone transfer for later studies, although it provides information that may be valuable which is not otherwise obtainable.

Based on this study, we concentrate our investigation of DNS related features on features involving the `NS`, `A`, `MX`, and `CNAME` records. We also saw that most operators follow the requirement of two name servers for a DNS zone, but do not provision extra redundancy beyond this requirement. In Chapter 4, we will see a provisioning practice used for malicious activity where this requirement is often greatly exceeded.

3

Data Sources

The research in this dissertation is data-driven, and relies on data from many different sources. In this chapter, we describe each of these data sources, broken down into data concerning likely benign host names, data concerning likely malicious host names and addresses, and data that does not distinguish between the two. Some are used for research presented in multiple chapters, while others are only used once in the dissertation. When data sources are used in multiple chapters, different snapshots of the data are used due to the times the studies were conducted. Not all data sources were available for all studies. In all cases, we leave the specifics of how we use the data to the chapters where the results based on it are presented.

3.1 Sources of Benign Hosts

`DMOZ` Our first source of benign hosts is the DMOZ Open Directory Project [15]. We use this data source in Chapters 5 and 8. The Open Directory Project is a large directory of user submitted Uniform Resource Locators (URLs). Its intended purpose is as a directory that can be used to locate relevant web sites, as a substitute for using a search engine. Links in the directory are categorized, although we do not take advantage of this categorization. Anyone can submit a link to be added to the directory. They will not be added until approved by an editor. The editors are volunteers, and must submit an application and be approved by senior editors. While it is possible that some malicious links may be approved if the editors are themselves

malicious, we do not believe this happens often, if at all. We have found only small overlap between the hosts from this source and our sources of malicious information, which may be due to the sites being hacked.

`Alexa` Another source of benign hosts are those from the Alexa Web Information Service [3]. We use this data source in Chapter 8. This service ranks the most popular pages on the Internet, based on the pages visited by users who install their toolbar, broken down into several categories. Although it is possible for some of these sites to be compromised, since the top sites listed in this service are those that are frequently visited, we expect that most are not malicious.

3.2 Sources of Neutral Hosts

Not all data sources contain only good URLs or only malicious ones. Those that contain a mix of good and bad in general do not provide a way to tell which are good or bad. Since we do not know if hosts in these data sources are good or bad we call them neutral.

`Zone_Files` Our main source for neutral hosts are zone files from several Top Level Domains (TLDs). Zone files are text files listing all DNS records directly contained in a domain. In the case of a TLD, the zone file must have name servers listed for all of the domains contained in the TLD. Thus, they can be used to obtain a list of all the domains in the TLD. In total, we use zone files from seven TLDs: `.asia`, `.biz`, `.com`, `.info`, `.mobi`, `.net`, and `.org` [2,16,68,74,85,109]. We use this data in Chapters 5 and 6. A notable deficiency of this data is that it does not include any Country Code Top Level Domain (ccTLD). Unfortunately, the operators of the largest few ccTLDs are not willing to share such data for research purposes.

`LocalDNS` A second source of neutral hosts consists of hosts visited by computers on our departmental network. While most are likely to be benign hosts, we do not have any reason to believe that nobody has visited malicious sites. To create this data set, we captured all the Domain Name System (DNS) queries issued on our departmental network, anonymized so we could not tell which computer was requesting which host names. We then extracted the host names contained in queries for IP addresses. We use this data in Chapter 8.

3.3 Sources of Malicious Hosts

Although we criticize blacklists for being reactive and incomplete, they are also the only available source for malicious hosts. Therefore, our malicious data sources mainly consist of several blacklists. We organize them below by the type of malicious activity they are meant to track.

3.3.1 Phishing Sites

We have used four different feeds of phishing URLs, provided by MarkMonitor [55], PhishTank [77], the Anti-Phishing Working Group (APWG) [5], and the Google Safe Browsing API [24].

`MarkMonitor` The URLs in MarkMonitor's feed are obtained from various large e-mail providers and Internet Service Providers (ISPs). To verify them, MarkMonitor passes them through a filter which performs URL and content analysis and determines the likelihood that the URL is pointing to a phishing site. For URLs with a high probability of being phishing URLs, MarkMonitor performs a manual check on the URL. The unique, positively identified URLs, are recorded along with the brand they target in their phishing feed. We use this feed in Chapter 4.

`PhishTank` PhishTank is a community-driven reporting system for phishing emails. The URLs in this feed are either user submitted or obtained via external feeds. The user-submitted URLs are voted upon for verification purposes. Submitted URLs must be verified by multiple individuals before they will be listed as a verified phishing page. All maintenance of this feed is handled on a volunteer basis. We use this feed in Chapters 4, 5, 6, and 7.

`APWG` The APWG is a group consisting of both research universities and several organizations from industry, all with an interest in tracking and stopping phishing. The URLs in this feed come from various sources including spam traps, consumer complaints, and brand owners. Some of the URLs in this feed are hand verified, some are verified by external communities, and some are automatically processed. While the feed allows us to determine which are which, we use all the URLs from the feed regardless of how they were verified. We use this feed in Chapters 4, 5, 6, and 7, and 8.

`SafeBrowsing` One additional blacklist of phishing hosts we make use of is the data available through the Google Safe Browsing API. This list, however, can not be used in the same manner as others. The only data available is hashes of phishing URLs, not the phishing URLs themselves. Because of this, we can not use this as a source of URLs, but it is still useful to check if a given URL is malicious. We use this data in Chapter 6.

3.3.2 Spam/Scam Sites

We use two lists of URLs contained in spam messages, one from SURBL [102], and the other from Support Intelligence [101]. Additionally, we supplement this with URLs extracted from spam messages received locally.

`SupportIntelligence` Our feed from Support Intelligence is updated every six hours. This feed contains URLs from spam as well as associated IP addresses. Although this feed is very large, certain scams are very heavily represented. At times over half of the URLs in the feed have pointed to a single scam. However, there is still sufficient variety in the feed to make it useful. We use this data in Chapters 6, and 7.

`SURBL` SURBL also collects domain names from URLs contained in spam. Although they typically only allow users to perform lookups on the domain names in their list, we have also arranged to receive the associated IP addresses from them as well. These IP addresses are those associated with the domain itself, and with the domain with `www` prepended. This feed is updated once per day. We use it in Chapters 5 and 7.

`LocalSpam` Our final source of scam URLs consists of URLs extracted from spam received at the Indiana University Computer Science email server. All emails are passed through the Sophos PureMessage [94] spam detection system. URLs are extracted from those marked as spam to create this data set. We receive the list of URLs daily. This data is anonymized, we do not know which URLs were destined to the same user, or any content of the email messages. We use this data set in Chapter 7.

3.3.3 Spam Senders

`SBL` IP address blacklisting can be used on mail servers to prevent compromised machines from sending mail directly. Spamhaus runs the most widely-used blacklist in this context, the SBL [98]. The SBL mainly contains IP addresses of machines verified as spam senders. Additionally, when known spam operations move to new hosts, corresponding entries are preemptively added. This list can be queried by mail servers when they receive connections to block known spammers. We obtain a copy of this blacklist every hour, and extract the IP addresses. We use this data set in Chapters 6 and 7.

3.3.4 Exploited Hosts

`XBL` Spamhaus also maintains a second blacklist, known as the XBL [96]. This list contains prefixes (often individual IP addresses) of hosts infected with exploits often used to send spam. This includes open proxies, computers infected with viruses that are known to send spam, and other exploits. This data is updated every half hour. We use it in Chapters 6 and 7.

3.3.5 Malware Distribution Hosts

Four of our data sources contain URLs of sites distributing malware. These are eSoft [18], the Clean-MX Viruswatch Mailing List [73], Malware Patrol [54], and the Google Safe Browsing API [24].

`CleanMX` The Clean-MX Viruswatch mailing list sends out URLs hosting malware, along with IP addresses and what specific malware was seen. The list formerly updated every 15 minutes, but now updates much more sporadically. We use this data source in Chapters 5, 6, and 7.

`MalwarePatrol` Malware patrol discovers URLs hosting malware through two methods. They accept user submissions, but they also crawl the web actively looking for malware hosting sites. Their list is updated hourly. We use it in Chapters 5, 6, and 7.

`eSoft` Similarly, eSoft also collects URLs of malware. Unlike the others, they also send out a sample of the malware along with the URL. However, we do not make use of this. We use this data

source in Chapters 5, 6, and 7.

`SafeBrowsing` The Google Safe Browsing API's malware feed is similar to its phishing feed, in that it does not contain URLs of malware directly. Instead it contains hashes, and is therefore not useful as a source of malicious URLs but can be used to check if a given URL is malicious. We use this data in Chapter 6.

3.3.6 Bot Command and Control

`ShadowServer` Botnets consist of groups of compromised machines used for malicious purposes on the Internet. Miscreants often use them for sending spam and for hosting phishing and scam sites. While we do not possess any direct sources of botnet IP addresses, many of the addresses in our other data sources are likely to be bots since bots are commonly used for malicious activity. However, botnets must get their instructions from their bot masters, often through command and control servers, which distribute orders. The ShadowServer Foundation [92] provides lists of botnet command and control servers along with their IP addresses. We use this data in Chapter 7.

4

Fast Flux

4.1 Introduction

A provisioning strategy known as *fast flux* has recently been popular among miscreants on the Internet. Miscreants use this technique to provide extreme availability of their fraud web sites in the face of take-down attempts. A web site exhibiting fast flux typically resolves to a large number of IP addresses, each with a low Time to Live (TTL), meaning that each is valid for use by clients for only a short period of time. Successive resolutions of the site often lead to a new set of IP addresses, increasing availability. At the same time, the short validity of these addresses ensures that the operators of these web sites can provide a new, up-to-date list of machines hosting these sites. Combined, these fast flux features help keep fraud campaigns afloat longer despite take-down efforts.

Fast flux comes in various flavors. The term is most commonly used to refer to the case when the web server name of a fraud-related site resolves to large number of IP addresses with short validity. In the rest of this chapter, when we refer to fast flux, it is meant in this sense. However, even the Domain Name System (DNS) server that leads clients around the world to the web server could exhibit flux. Further, this phenomenon could continue all the way into the DNS hierarchy of the domain associated with fraud. To distinguish between the case when a web server exhibits flux from one where its DNS servers exhibit flux, the latter activity is referred to as *DNS flux*. The term *double flux* refers to the case where fast flux and DNS flux are occurring together.

Previous work has shown that many scam web sites utilize fast flux [30, 47]. However, little is known about the use of flux in phishing campaigns. In this chapter, we study fast flux, DNS flux, and double flux in the context of phishing. Although we do not study flux in the context of malware sites, they would benefit from using it as well, so it may also be a useful feature to identifying them.

We use an Support Vector Machine (SVM) [9] classifier on real-world data to produce models which can identify all types of flux. Using our mechanism, we find that phishing campaigns utilize fast flux, as well as DNS flux, although less often than web sites connected to spam. We also find that double flux helps ensure the longevity of phishing campaigns more than just fast flux. Our findings suggest that the ability to detect flux will be useful in fighting various kinds of cybercrime.

We go beyond simply examining the prevalence of flux in phishing. In contrast to prior work which simply checks if flux can be detected using a list of features [30,80], we perform feature selection to determine which of the features we examine is actually helpful in identifying flux. Additionally, we examine the practicality of detecting flux in real time, without causing significant performance degradation of web browsing. Finally, in order to determine the true size of fast flux campaigns, we perform clustering on the fast flux hosts. This exercise reveals several interesting insights about the relationships among fluxing machines.

The key results of the work in this chapter are:

- Prevalence: 11% of the web servers hosting phishing sites exhibit fast flux. 62% of the DNS servers in use by phishing sites exhibit DNS flux. In fact, 78% of the web servers exhibiting fast flux were pointed to by DNS servers that themselves exhibited DNS flux. Thus, most of the phishing web servers that exhibited fast flux were also a part of a double flux infrastructure.

- Clusters of Fluxing Machines: When clustering host names exhibiting fast flux based on 50% commonality of the IP addresses they point to, we find just 12 clusters of just fast flux host names and 25 of DNS flux. This suggests that many of the fluxing hosts are working together. We find that hosts within a cluster are named using similar patterns, which also hold across some of the clusters.

- Accuracy: Unfortunately, identifying DNS flux is harder than identifying fast flux. While fast

flux can usually be identified using the set of addresses returned from a single DNS resolution, DNS flux is harder to identify, requiring at least 10 resolutions for reasonable accuracy.

- Performance: Classifying a host name as fluxing or not would add 0.78% overhead to a typical DNS resolution, implying that the classification proposed in this paper could be realistically implemented at the client.

4.2 Data Collection

To collect data to investigate flux, we use the three near real time feeds of phishing Uniform Resource Locators (URLs) described in Chapter 3: `MarkMonitor`, `PhishTank`, and `APWG`. Each URL in the feeds belongs to a phishing web site. From the URLs in these feeds, we extract a list of web server host names. We then perform several DNS queries on each host name. We look up the `A` records corresponding to the host name, and the `NS` records for each domain and subdomain contained in the host name. Then, we also look up the `A` records for each DNS server for this host discovered in the `NS` lookups.

For example, if the feed contained the host name `www.xyz.example.com`, we would look up `A` records for `www.xyz.example.com`. We would also look up the `NS` records for `www.xyz.example.com`, `xyz.example.com`, and `example.com`, Finally, we would look up the `A` records corresponding to each of the `NS` records returned from the three `NS` requests.

We perform each DNS resolution periodically, once every 15 minutes. We choose this interval because the caching duration for most fluxing records is shorter than 15 minutes. This avoids receiving answers for queries about fluxing domains from the local cache. We continue attempting DNS resolutions for each host name as long as they are in our data feed or the resolutions are succeeding. We stop attempting the resolve a host name if the host name is no longer in our data feed and the resolutions have failed for one full day.

In order to accurately infer the presence of flux, we collect several other pieces of information for each IP address discovered above. First, we perform geo-location using the IP2Location [29] software. This helps in inferring which countries the phishing infrastructure is located in. Additionally, we

	Web Server	DNS Server
Resolved host names	15,547	77,568
IP addresses	15,230	26,214
Autonomous systems	1,851	4,602
BGP prefixes	5,212	10,605
Countries	106	128

Table 4.1: Overview of data used for fast flux detection

use Border Gateway Protocol (BGP) routing tables from routeviews [76] to determine the BGP prefix and Autonomous System (AS) for each IP address. All of these can be used as indications of how far apart the set of IP addresses corresponding to a host name are from each other, either geographically or in terms of network structure. We expect that legitimate web sites using multiple IP addresses will have the IP addresses grouped together in the same AS, while malicious sites will not due to use of botnets.

4.2.1 Overview of Collected Data

We collect data for a period of one month (31 days) starting August 1, 2008. The feeds contain 53,154 URLs. Sometimes, multiple URLs in the feed contain the same host name. These URLs correspond to 30,450 host names. We were able to resolve 15,547 (51.1%) of these, details of which are shown in Table 4.1. The rest were likely taken down by the time they appeared in our feed. The resolvable host names lead to 15,230 IP addresses. Even though some web servers are hosted on many IP addresses, many share some of their IP addresses. The result is that the web server names are close in number to the IP addresses.

Also shown in Table 4.1, the 77,568 DNS server names used by the phishing sites correspond to just 26,214 IP addresses. In many cases the web server name did not resolve, so we could not determine if the site used fast flux, but we were able to obtain `NS` records. If we exclude these cases, including only the ones where the host of the web site was still available at the time of our measurements, there were 46,889 DNS server host names with 7,529 IP addresses.

4.3 Detecting Flux

Our goal is to identify flux in phishing data. Having collected the data described above, we now apply a machine learning algorithm to this problem.

4.3.1 Methodology

Given the IP addresses from one or more resolutions of a host name and associated DNS servers, we would like to determine whether fast flux, DNS flux, or double flux is occurring. Toward this goal, we apply SVMs, a class of machine learning algorithms. SVMs produce a data partitioning in the form of a hyperplane which simultaneously minimizes the classification error and maximizes the geometric margin between the partitions. This hyperplane can be thought of as the "line" between the partitions which is "farthest" from all of the partitions simultaneously. Upon training an SVM on a given set of data, one gets a model which separates the data into two classes using a partition that minimizes error and maximizes the margin between the classes. This model can be used to rapidly classify new data points.

The SVMs operate on vectors each representing a data point, containing the value for all the features, and the appropriate class if the data is being used for training. Making use of a kernel function, data is mapped into a higher dimensional space, where the optimal hyperplane separation is found via optimization methods. The particular library used for this work, `libsvm` [9], uses the radial basis function (RBF) kernel by default. For the purposes of this work, the defaults within `libsvm` were chosen in all cases, and accepted best practices were used for scaling the data, and training and validating the model.

Using the parameters described subsequently in Section 4.3.2 we train two SVMs, one to determine fast flux, and the other to determine DNS flux. Determining double flux is straightforward once fast flux and DNS flux are detected. We select 10% of the data at random to train the SVM. This training includes a ten-fold cross validation to ensure a consistent model, unbiased by any particular subset of the data. In order to train the SVM, we need pre-classified data points. For this pre-classification, we apply a heuristic to this 10% based on the observation that fast flux host names

have an ever-increasing number of IP addresses returned on successive resolutions. Over time, the number of IP addresses greatly outstrips all cases where flux is not present [30].

Once we have completed the pre-classification, we ensure that there is a good mix of fluxing and non-fluxing hosts in the training data. It is important to be careful not to misclassify sites using Content Distribution Networks (CDNs), since similar to fast flux hosts, they are known to return a relatively large number of IP addresses corresponding to a host name, each with a short validity. However, CDNs generally do not return as large a number of IP addresses as flux networks do, and the number of IP addresses returned by CDNs generally does not grow as quickly with the number of resolutions performed. In order to ensure that our models handle the CDN case correctly, we add to the training data 13 popular, legitimate sites in the Internet that use CDNs. The end result of training the SVMs is a model which can be used to partition the data into the two classes: "flux"/"not flux". Training a model for DNS flux follows an identical process to training for fast flux. In this case, the heuristic checks if the DNS servers have an ever-increasing number of IP addresses built up from successive resolutions.

4.3.2 Parameters Used

To determine the occurrence of each type of flux, we explore a range of parameters. Some of these are common to other works in the area [8,30,71,80] though these works only explore them in the context of fast flux and do not investigate DNS flux or double flux. There are two major differences in how we construct our model as compared to these works. The first difference stems from our choosing to only use parameters that can be easily and accurately derived. Thus, we avoid parameters based on the Internet *whois* database, which contains owner and other related information about domains but is sometimes unreliable [34]. The second difference is that we strive to derive the smallest set of parameters required for inferring each type of flux with highest accuracy. Specifically, we consider the following parameters:

- **Number of IP addresses (n_{IP})**: The biggest indicator of any kind of flux is the total number of IP addresses obtained as the result of resolving a host name. Thus, we use the total number of IP addresses obtained from all DNS resolutions of a host name as our first

parameter. However, non-fluxing servers may have large numbers of IP addresses for other reasons, so other parameters must be considered.

- **Number of ASes to which those IP addresses belong (n_{AS})**: Generally, all the IP addresses corresponding to a host name co-exist in the same AS, as the host is maintained by a single administrative entity. Even for CDNs, results from a single vantage point exhibit the same behavior. However, this is not true for fluxing hosts, simply because the bot armies of compromised machines that they host their sites on typically belong to multiple ASes. Thus, if the resolved IP addresses belong to many ASes, we take this to be a possible indication of flux.

- **Number of prefixes to which those IP addresses belong (n_p)**: BGP Prefixes are continuous groups of IP addresses announced from a single AS. The reasoning behind looking at IP prefixes that the resolved IP addresses belong to as an indication of flux is similar to that behind looking at the number of ASes. The IP addresses corresponding to legitimate hosts usually belong to a handful of BGP prefixes per host name while this is unlikely to be true for networks exhibiting flux. Thus, if a host name belongs to many BGP prefixes, this is a possible indication that it is fluxing.

- **Number of countries to which those IP addresses belong (n_c)**: Even though many domains are hosted in multiple countries, individual hosts reside in a single country in most cases. In fact, the hosts belonging to a particular country code top level domain (ccTLD) would in most cases be located on IP addresses physically residing within that country. Due to this fact, if a host name belongs to multiple countries, we take it to be a possible indication of flux.

- **Number of DNS servers corresponding to web servers (n_{NS})**: Typical web servers generally are associated with only a handful of DNS servers. This number is typically two for most web servers in the Internet, as seen in Chapter 2. Web servers exhibiting fast flux typically have many more associated DNS servers [30]. Thus, we take a large number of DNS servers corresponding to a web server as a possible indication of fast flux. Notice that this parameter is not applicable to DNS flux detection.

- **Short average TTL (TTL)**: Fluxing hosts typically use a shorter average TTL on their IP addresses than legitimate hosts. This is because miscreants want to avoid the possibility of client resolvers caching them. The shorter the TTL, the faster a host can change its `A` records. Thus, we consider a short TTL as a possible indication of flux. We use TTL as a binary parameter in our model, using a threshold of 10 minutes.

4.4 Prevalence of Flux

We are now in a position to answer the question: How prevalent are fast flux, DNS flux, and double flux in phishing?

First, to detect fast flux, we trained an SVM on 10% of the data using the complete set of parameters discussed in Section 4.3.2. We then applied the trained SVM to the remaining 90% of the data. We found that 11.4% (1,733) of the web servers that were alive at the time of our measurements exhibited fast flux. This percentage was recently observed to be 30% in the context of web servers hosting scam sites pointed to in spam [30]. Interestingly, these 11.4% web server names corresponded to 45.5% of the phishing IP addresses in our data. Clearly, the fluxing web servers cycle through a large number of IP addresses.

Next, as we did for fast flux, we trained a SVM classifier on 10% of the data to infer the presence of DNS flux in the remaining data. In this case, we used all the parameters except n_{NS}, the number of DNS servers corresponding to a web server. This parameter obviously does not apply to DNS flux. For web server host names which contain multiple sub-domains, we looked for DNS flux at all levels up to the second level domain name, as described in Section 4.3. We found that 61.7% (47,889) of DNS servers exhibited DNS flux. Most of the phishing host names with DNS flux, 67.7%, had only their immediate DNS server fluxing. The remaining 32.3% had DNS server host names fluxing at multiple levels of the DNS hierarchy. This protects them against clean-up of any of the hosts they uses as web servers or any of those they are using as DNS servers.

Clearly, more DNS servers flux than the web servers, both in absolute numbers and in percentage. It is counter-intuitive that more DNS servers should flux than even the web servers that they point

to. To gain a better insight, we looked at the IP addresses corresponding to the fluxing DNS servers. Surprisingly, there were only 904, indicating that while many DNS server names exhibit flux, they correspond to far fewer actual machines.

Finally, we looked for the presence of double flux in our data. We found that 77.6% of the fluxing web servers were part of a double flux network. This implies that most of the times when phishing infrastructure fluxes, it fluxes in all possible ways. In fact, in 98.8% of the double-flux cases, the associated DNS servers were fluxing at multiple levels of the DNS hierarchy. These percentages imply that a large percentage of the phishing campaigns are very well-provisioned against take-down efforts, for they exhibit flux both at the web server granularity as well at various levels of DNS-server hierarchy.

4.4.1 Impact of Parameter Set

Our initial SVM model for fast flux was created with all six parameters, while the one for DNS flux used five. We now test which parameters offer the best accuracy without being redundant. We produce SVM models with all combinations of parameters, and compare their results to the full model.

Parameters for Detecting Fast Flux

We begin by examining how web servers with and without fast flux fare for each of the six parameters discussed in Section 4.3.2. Specifically, in Figure 4.1, we show the Cumulative Distribution Function (CDF) of fast fluxing and non-fast fluxing web servers for each of the parameters (with the exception of TTL, for which we show a Complementary CDF (CCDF)). We see that for each parameter, there is a very clear difference in the lines plotted for hosts with fast flux and those without it. Approximately 80% of non-fluxing web servers have a single IP address, belong to a single BGP prefix, and a single AS, while 90% belong to a single country. Additionally, 85% have at most two DNS servers, and over 60% have an average TTL of longer than 10 minutes. In contrast, among the web servers that exhibit fast flux, only about 2% are in a single AS, prefix, or country, fewer than 20% have two or fewer DNS servers, and fewer than 20% have average TTLs of longer than 10 minutes. Clearly, each

of the parameters offer promise in helping the detection of fast flux. We now investigate which ones identify the most fluxing servers (low false negatives) without misclassifying non-fluxers (low false positives).

In order to determine the least set of useful parameters, we create SVMs to classify phishing web servers with all the parameters, and sequentially peel off a parameter at a time. If there is little to no change between the presence of the parameter and absence of the parameter, it was deemed to be of no additional discriminatory value over the parameters still present. In this fashion, we can determine the minimum set of parameters necessary for classification with good accuracy. A comparison of models with 3, 4, or 5 features to the full 6 feature model is shown in Table 4.2. Models with fewer features in general performed worse than those shown.

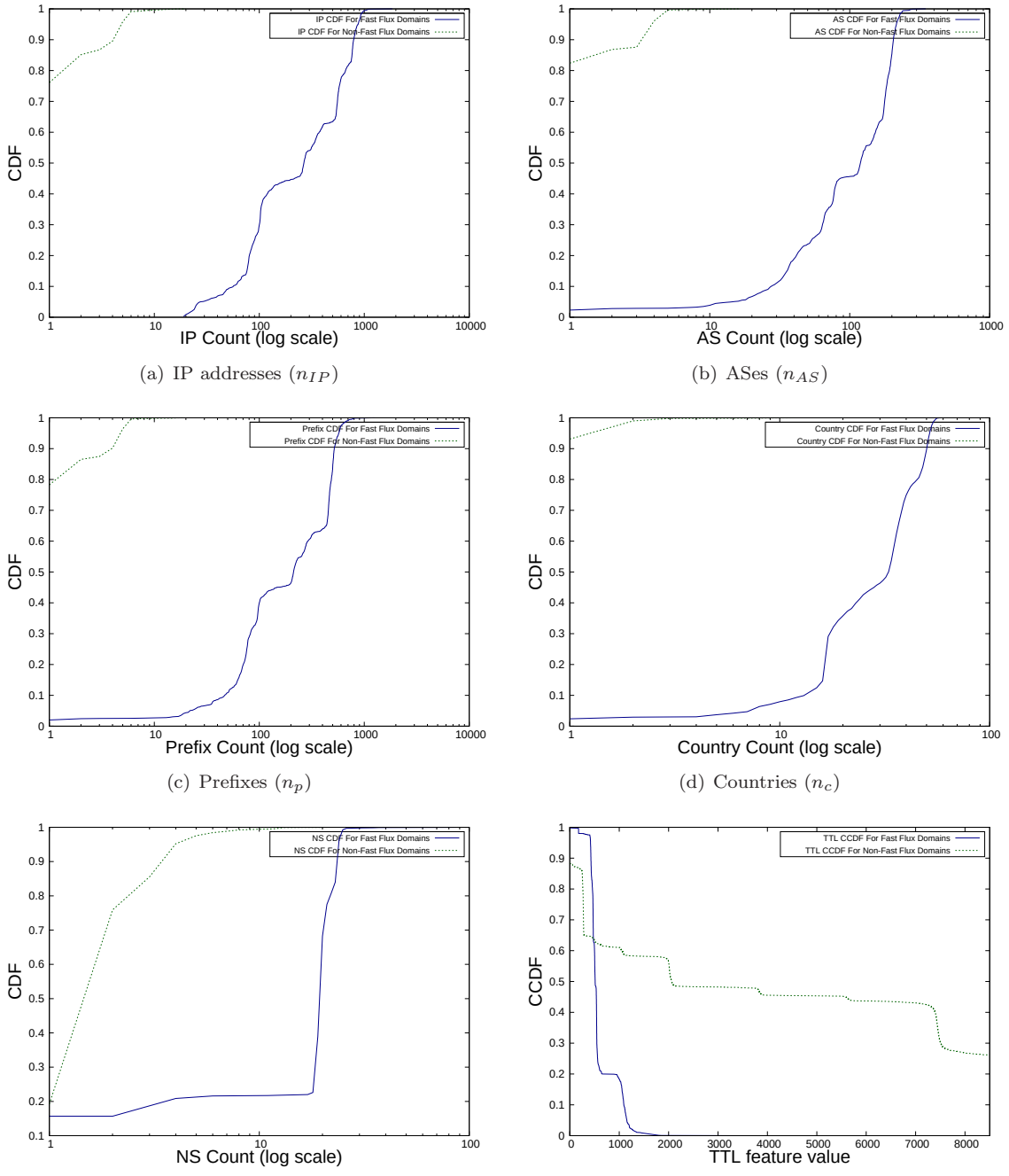
We find that the combination of four parameters, $\{n_{IP},\ n_{AS},\ n_p,\ n_c\}$ produces results with relatively few misclassifications. It has only three false negatives (fluxing hosts classified as non-fluxing) out of the 1,773 total fluxing hosts, and still maintains low false positives (non-fluxing hosts classified as fluxing). We conclude that all six parameters are not required to detect fast flux. This four parameter model is used for results throughout this chapter.

Parameters for Detecting DNS Flux

We repeat the same process we used to investigate which parameters were useful in fast flux for DNS flux. We find that several models do well. Somewhat surprisingly, the classifications with a very simple model are almost as good as with the full one. The single parameter model with just n_{IP} has only 2 false negatives and 10 false positives as compared to the full model. We conclude that just n_{IP} is sufficient to detect DNS flux.

4.4.2 How Many DNS Resolutions Are Enough?

One of the identifying characteristics of fast flux and DNS flux is that a new set of IP addresses may be returned at each resolution. Typically, as the number of resolutions increase, so do the total number of IP addresses seen. This implies that having results of multiple DNS resolutions



(a) IP addresses (n_{IP})

(b) ASes (n_{AS})

(c) Prefixes (n_p)

(d) Countries (n_c)

(e) DNS servers (n_{NS})

(f) Average TTL (TTL)

Figure 4.1: Cumulative distribution function of each parameter used in fast flux detection for fluxing and non-fluxing web servers

Feature List	Flux	False Neg.	False Pos.
All features	1,773	–	–
n_{ASN}, n_p, n_c, n_{NS}, TTL	1724	51	0
n_{IP}, n_p, n_c, n_{NS}, TTL	1726	50	1
n_{IP}, n_{ASN}, n_c, n_{NS}, TTL	1699	76	0
n_{IP}, n_{ASN}, n_p, n_{NS}, TTL	1792	7	24
n_{IP}, n_{ASN}, n_p, n_c, TTL	1784	0	9
n_{IP}, n_{ASN}, n_p, n_c, n_{NS}	1783	1	9
n_p, n_c, n_{NS}, TTL	1646	129	0
n_{ASN}, n_c, n_{NS}, TTL	1623	152	0
n_{ASN}, n_p, n_{NS}, TTL	1814	0	39
n_{ASN}, n_p, n_c, TTL	1726	49	0
n_{ASN}, n_p, n_c, n_{NS}	1336	466	27
n_{IP}, n_c, n_{NS}, TTL	1682	93	0
n_{IP}, n_p, n_{NS}, TTL	2293	0	518
n_{IP}, n_p, n_c, TTL	1775	11	11
n_{IP}, n_p, n_c, n_{NS}	1135	649	9
n_{IP}, n_{ASN}, n_{NS}, TTL	1826	0	51
n_{IP}, n_{ASN}, n_c, TTL	1712	69	6
n_{IP}, n_{ASN}, n_c, n_{NS}	1710	74	9
n_{IP}, n_{ASN}, n_p, TTL	1788	1	14
n_{IP}, n_{ASN}, n_p, n_{NS}	1781	7	13
n_{IP}, n_{ASN}, n_p, n_c	1781	3	9
n_c, n_{NS}, TTL	0	1775	0
n_p, n_{NS}, TTL	1764	14	3
n_p, n_c, TTL	1674	101	0
n_p, n_c, n_{NS}	1712	63	0
n_{ASN}, n_{NS}, TTL	1688	87	0
n_{ASN}, n_c, TTL	1618	157	0
n_{ASN}, n_c, n_{NS}	1607	168	0
n_{ASN}, n_p, TTL	1797	1	23
n_{ASN}, n_p, n_{NS}	3100	0	1325
n_{ASN}, n_p, n_c	1816	0	41
n_{IP}, n_{NS}, TTL	1849	5	79
n_{IP}, n_c, TTL	1682	93	0
n_{IP}, n_c, n_{NS}	1715	66	6
n_{IP}, n_p, TTL	1777	48	50
n_{IP}, n_p, n_{NS}	1830	1	56
n_{IP}, n_p, n_c	1765	21	11
n_{IP}, n_{ASN}, TTL	1804	0	29
n_{IP}, n_{ASN}, n_{NS}	1794	1	20
n_{IP}, n_{ASN}, n_c	1697	84	6
n_{IP}, n_{ASN}, n_p	1773	11	9

Table 4.2: Impact of parameters used on false positives and negatives in classifying fast flux

is useful in accurately identifying the presence of flux. In this section, we investigate the issue of how many DNS resolutions are sufficient to provide enough information to accurately distinguish fluxing host names from non-fluxing ones. The answer to this question has important implications for the applicability of our technique at clients. Requiring fewer, possibly one, DNS resolutions to infer flux can help the DNS resolver of the client protect the client from visiting a malicious site. If the technique requires more than one, then it is higher overhead, and the detection is no longer practical in real time for the client to use as a defense mechanism.

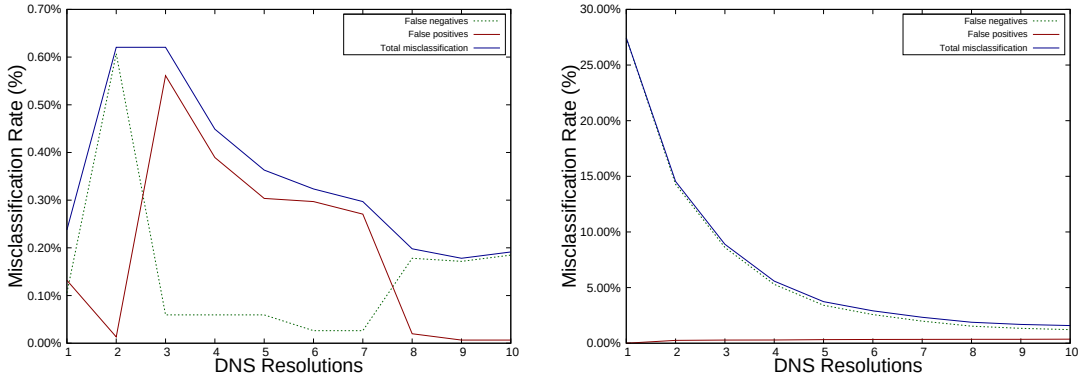
Resolutions for Inferring Fast Flux

In order to determine the marginal returns on successive lookups, we constructed SVM models using the best (minimal) parameter set for fast flux, $\{n_{IP}, n_{AS}, n_p, n_c\}$, and data obtained for each from one to ten DNS lookups. The output of each model was compared to the model constructed in Section 4.4.1, which used all resolutions performed for each web server, with all six parameters. Specifically, we look for false positive rate – classification of good hosts as fast flux, when they were not – as well as false negative rate – classification of fast flux web servers as non-fast flux, for each model.

As seen in Figure 4.2(a), only a very small benefit is gained from using multiple resolutions. While the misclassification appears to be very erratic, the maximum misclassification rate is approximately 0.64%. Among the small number of misclassifications we do see for a single DNS resolution, there are more false negatives than false positives. This is desirable because we wish to avoid penalizing good hosts. Because even with one DNS resolution, misclasifications are low, we conclude that a single DNS resolution is sufficient to classify a given web server name as fast flux or not. This further implies that this method could easily be integrated in an anti-phishing filter at the client itself or its DNS resolver.

Resolutions for Inferring DNS Flux

Here, we investigate if multiple resolutions are required for identifying DNS flux. The outcome here is quite a bit different than that of fast flux. Specifically, Figure 4.2(b) shows that with only one DNS



(a) fast-flux

(b) DNS-flux

Figure 4.2: Effect of varying the number of DNS resolutions on the accuracy of fast flux and DNS flux identification

resolution on DNS servers, we see over 25% misclassifications. This is far higher than the maximum for fast flux. Even with ten DNS resolutions, we still see 2% misclassification, an order of magnitude more misclassification than in the case of fast flux. Fortunately, most of the misclassifications are false negatives, not false positives, implying that good DNS servers are rarely flagged as fluxing while fluxing DNS servers are missed more often than desired. We conclude that identifying DNS flux with fewer resolutions is significantly more difficult than identifying fast flux.

The obvious question is why DNS flux is so much more difficult to identify than fast flux. We find two reasons for this. First, because secondary DNS servers are often provisioned as backups, a legitimate DNS server host name in our data is much more likely than a web server host name to have multiple IP addresses in the same country, averaging two IP addresses per country instead of one. For the same reason, it is also somewhat more likely to have multiple IP addresses in the same AS and prefix. Second, while we see a large difference in the median number of IP addresses returned for non-fast flux web servers and fast-flux web servers, 1 versus 14, this difference is not present for fluxing and non-fluxing DNS servers. Instead, the median number of IP address records returned for both of these is 1, indicating that the fluxing DNS servers are only fluxing sometimes, not every time we look them up. These differences are likely the major causes of the difficulty identifying DNS flux domains with a low number of lookups.

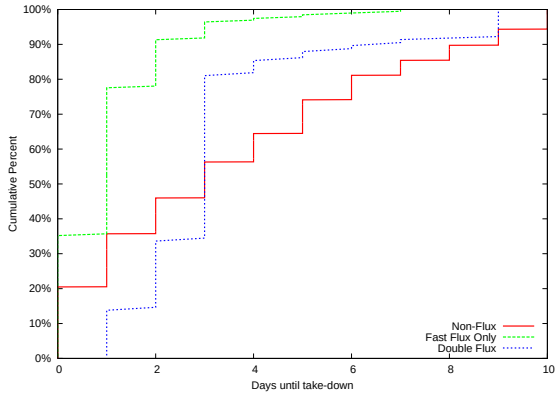


Figure 4.3: Comparison of the lifetimes of double flux, fast flux only, and non-fluxing phishing domains lasting 10 or fewer days

4.5 Flux and Fraud Longevity

Given that some phishing campaigns set up their infrastructure with fluxing hosts, an obvious question that comes to mind is: Does flux help with the longevity of fraud campaigns? We look at the length of phishing campaigns with and without flux to gain an insight into this question.

In Figure 4.3, we show a comparison of lifetimes of *domains* associated with phishing. We focus on domains that last shorter than 10 days after being blacklisted. Though this cut-off is somewhat arbitrary, the reasoning behind ruling out domains that last longer is to avoid being biased by those domains that remain unnoticed by those performing take-downs for a long period of time, perhaps because they are less effective. The phishing web servers in such domains will continue to operate irrespective of whether or not they exhibit any form of flux. We divide the domains in Figure 4.3 into three categories: those whose servers do not exhibit any kind of flux, those whose web servers exhibit fast flux but no DNS flux exists, and those that exhibit both fast flux and DNS flux (double flux).

The curves in Figure 4.3 are somewhat of a surprise: domains with just fast flux last for shorter duration than those without any form of flux. Specifically, while 65% of the phishing domains without flux last longer than a day, only 20% of domains with just fast flux manage to live more than a day. On the other hand, double flux seems to help with fraud longevity, for 83% of domains with double flux survive more than a day. Clearly, when deciding to provision the phishing infrastructure with

flux, it pays more to provision it with double flux than just fast flux.

In reasoning why domains with just fast flux fare worse than those without any form of flux, we recognize the recent attention that fast flux has received from commercial enterprises [8, 71]. This could be leading to faster detection and take-down of domains that exhibit fast-flux characteristics. DNS flux on the other hand has been described [104] but not studied, other than in this article. This may help those using DNS flux. Given that removing the DNS servers for a domain is a possible way of taking it down, a large number of DNS servers that frequently change and are not under the same administrative control make take-down more difficult.

Figure 4.3 shows another interesting aspect: While double flux provides benefits to the longevity of phishing campaigns in the short term, it seems to be detrimental in the long term. Specifically, only 20% of phishing domains with double flux last for more three days while close to half of those without any form of flux last more than three days. The question becomes: Why does double flux become detrimental in the long run? We conjecture that the difficulty of taking down double flux domains keeps them up in the short term, but the fast-flux behavior attracts more attention in the long run.

4.6 Top Fluxers

To get a better idea of the activity surrounding a flux campaign, we now specifically examine the largest fluxing operations in our data. Specifically, to determine which fluxing hosts belong to the same groups of miscreants, we perform clustering of the web server host names based on their IP addresses. We perform this clustering separately for the double flux web servers and those with just fast flux. For the double flux, we also cluster the name server host names. Toward this goal, we first compute the *Jaccard distance* [38] between each pair of host names by looking at the IP addresses they have in common. Let IP_{h_i} and IP_{h_j} be the sets of IP addresses of hosts h_i and h_j. Then Jaccard distance for each pair of hosts, h_i and h_j, is given by: $J(IP_{h_i}, IP_{h_j}) = \frac{|IP_{h_i} \cup IP_{h_j}| - |IP_{h_i} \cap IP_{h_j}|}{|IP_{h_i} \cup IP_{h_j}|}$

We then apply the *single-linkage clustering* [38] technique to examine which host names end up in the same cluster. The single-linkage clustering uses distance, $D(X, Y)$, between each pair of

clusters, X and Y, and hierarchically combines the clusters with the smallest distance (or greatest commonality, where commonality is $1 - D(X, Y)$). In our case, we start with each host belonging to its own cluster and use minimum Jaccard distance between hosts in separate clusters to hierarchically combine the clusters using this technique. Depending on the threshold we use to combine clusters, we end up with a variable number of clusters.

4.6.1 Characteristics of Just-Fast-Flux Networks

We find 22% of fast flux host names have just fast flux without DNS flux. We clustered these web server host names based on their IP addresses. Table 4.3 shows the number of clusters we get when we combine clusters based on various levels of commonality. A small level of commonality may exist by coincidence, or by the same host being infected more than once and thus a member of multiple botnets. Any more than a few IP addresses in common probably does indicate a real connection between the host names. While a lower threshold would likely still be showing actual relationships, we conservatively focus the remaining analysis on the clusters at the 50% commonality level, which has 59 clusters.

Commonality	$> 0\%$	10%	25%	50%	75%	90%
Clusters	18	21	34	59	92	198

Table 4.3: Number of clusters of just fast flux host names based on the percentage of IP addresses they share

We first focus on the host names in the largest cluster. This cluster contained 83 host names. All of the host names were very similar and targeted three different banks. An anonymized example is `[bank].webbiz.wirebiz.globalupdate.certificateupdate.updatesessioniotzsisc2c226fo .configlogin.comreportid.[domain].com`, where in the real host name, [bank] and [domain] are replaced with the name of a bank and a domain name.

The other names in this cluster are all similar to this example in several ways. All of them contain a large number of labels (between nine and eleven), with some individual labels in common. All have `updatesession` or `Updatesession` followed by 15 random characters in the same position. All of them also have the bank name as the first or second label. We see a few host names in each

domain name. The rest of the labels between the domain name and `updatesession`, and one label after it, change for each host name. The remaining labels are always the same for each host name which is targeting the same bank. This shows that the IP addresses are not the only relation between the hosts names in a cluster, they use similar naming conventions as well.

This single large cluster is not the only one which follows a similar pattern. The four largest clusters and the 7th largest, along with a few smaller ones, all follow similar patterns. The 5th largest follows a similar pattern, only with `CommunityID` follows by nine digits instead of `Updatesession`. These clusters together account for 67% of the just fast flux domain names. Based on the large number of similarities in these clusters, it is likely that all of the just fast flux phishing falling into these clusters is due to a single group or a single kit.

These clusters share other characteristics as well. Normally, each label in a host name represents a level of the DNS hierarchy, so with 9-11 labels, these host names would be expected to have 9-11 levels of DNS servers. Instead we see something quite different. They are not declaring different DNS servers at each level of the hierarchy. In fact, the only level that has a DNS server is the domain name itself. Beyond the domain name, the entire remaining 7-9 labels are being served out of the DNS server at that level. This would ease administration, since under this system, many fewer DNS records would have to be created to make a new host name with a different set of the 7-9 labels beyond the domain name. Each cluster also only uses a small number of DNS servers, always named ns1 and ns2, in 1 to 5 domains per cluster.

4.6.2 Characteristics of Double-Flux Networks

The remaining 78% of fast flux host names also exhibit DNS flux, together known as double flux. To see the relationships among web servers, with double flux, we perform the same clustering we did for those with just fast flux. The number of clusters at various levels of similarity is shown in Table 4.4. Looking in slightly more detail at the case where we require a commonality of at least 50% between a pair of web server host names in different clusters in order to combine the clusters, we see a large degree of infrastructure sharing, with only 12 clusters. The largest three clusters account for 45%, 23% and 23% of the double flux host names, or 1/3, 1/6, and 1/6 of all fast flux host names.

Commonality	> 0%	10%	25%	50%	75%	90%
Clusters	1	3	5	12	158	1010

Table 4.4: Number of clusters of double flux host names based on the percentage of IP addresses they share

We begin by looking at the largest double flux cluster. This cluster contained 615 host names. All of the host names in this cluster followed a single pattern. An anonymized example is `ww9.[bank].com.[domain].su`. `[bank]` represents a real bank name and [domain] represents a domain name.

The hosts in this cluster are all in the `.su` Top Level Domain (TLD). The `[domain]` label takes two forms. It is often three to four letters followed by a number. When it is not that, then it is always two words put together with one of the words being update, verify, or confirm. The label which in our example is `com`, appears to be an attempt to trick the user by making it look like that label and everything to the left of it is a legitimate domain name. The only variation we see in this label within this cluster is sometimes `co.uk` us used in place of `com`. In this cluster `[bank]` is always one of three different banks.

We often see several very similar host names appear, identical except for the leftmost label, and often with only the digit in this label changed (such as `ww9`, `ww8`, `ww7`...). Much of the time, we see a set of names all identical except for this digit appear in our data feeds all at the same time. However, sometimes we see the variants with some digits appear on one day, but some are missing from the set. Those with the missing digits appear in our feeds several days later. We conjecture that in this case all these host names likely went into use at the same time, but some were not detected as phishing and submitted to our feeds until later. Sometimes, the missing variants never make it to our feeds at all. In this case, we conjecture that the missing variants likely really do exist and are being used, they have just not been detected yet. These patterns may be used to generalize entries in blacklists.

We also see a distinct pattern in the names of DNS servers used by the hosts in this cluster. All of the host names have 19 or 20 DNS servers with names consisting of ns{1-20} appended to the front of the exact host name. Further, this pattern is repeated at each level of the domain hierarchy. This

makes it appear that they have a highly redundant DNS system, but further investigation reveals that this is not the case. We first look at the IP addresses of each DNS server for the host name, and compare to the set of IP addresses for all the DNS servers for the name. Nearly all of the servers ns{1-20} for each host name have 90% commonality with each other with regards to the set of IP addresses they point to. Put another way, they all refer to the same sets of physical servers.

We also look at consistency up the levels of the DNS hierarchy. We compare DNS servers' IP addresses for all the DNS servers for a host name with those of the level above, and find all the host names have at least 61% of DNS server IP addresses in common with the level above, and 49% use all the same IP addresses as the level above. This indicates the same DNS infrastructure is used at every level, but with different host names assigned to it. This is perhaps to avoid detrimental effects of individual DNS server or even all of the DNS servers for a host name getting blacklisted. Since the different names really are the same server, then if the DNS server for the level above is asked for the IP address of the phishing site, it will still be able to answer.

We now move on the other double flux clusters. Looking at the naming conventions, we find that the host names in all except for one of the double flux clusters follow a very similar naming pattern, the only changes being the addition of one more bank, and changes in the TLD being used. For example, the second largest cluster targets all four banks seen, and the domains are all in the `.ru` (Russia) or `.co.uk` (United Kingdom) TLDs. Other behaviors in these clusters are also similar to the largest one, such as how they provision their DNS servers. They all use the same scheme where every host name uses 20 DNS servers with names consisting of ns1-ns20 prefixed to the host name, and all pointing to similar sets of actual servers. Together, these similarities suggest that 99.5% of the double flux we see is due to a single group or a single kit.

4.7 Discussion

We have seen that it is practical to detect fast flux using a single DNS resolution. This could be done either at end hosts, or by a centralized system proactively searching for malicious sites. We also see that miscreants taking advantage of flux are using a large number of host names. The use of many IP addresses and many host names makes these attacks difficult to completely block using current

methods. However, since they all use flux our method of identifying them using infrastructure features would be far more effective.

We analyzed the performance of the necessary operations to detect flux on a computer with a 2.4GHz Intel Core 2 Duo processor and 4GB of RAM. We compiled the code with the gcc compiler using high optimization levels. Timings were conducted on $4,770$ host names and no caching was used to improve performance in the face of IP address overlap. Given the sheer amount of overlap in IP addresses among hosts, such a cache should improve performance. The time required for classification alone was 15.09μs per classification. More time was used building the data point to classify, which involved the IP to AS mapping and geo-location. When the time to build the data points was included, the average time rose to 1.286ms per host name. To put these numbers in perspective, we conducted DNS resolutions. The goal was to compare the extra overhead flux-detection calculations would incur on median DNS resolution time. We found the median DNS resolution time to be 164ms over 2.9 million unique DNS resolutions. We conclude that a check for flux on each DNS resolution would then typically add only 0.78% overhead on top of the DNS resolution that must be made anyway to reach the web site.

Retraining the SVM is an important consideration for our system in practice. While we never retrain our SVM, we only analyze a month of data. In a practical system, retraining periodically would be necessary to keep up with changes in fast-flux behavior, especially attempts to avoid triggering this system. This retraining would likely not have to be done often, so it would add little overhead. If these checks were implemented at the end user system, the system would not necessarily have to deal with retraining. Instead, training could be done centrally, with a new SVM model then downloaded periodically by end systems.

5

DNS Wildcards

5.1 Introduction

In this chapter, we examine *wildcards*, one of the original features of Domain Name System (DNS) defined in the original standard. The role of wildcards in DNS is a many to one mapping, allowing all names within a single domain or subdomain to map to a single value, for example one IP address. Recently, Netcraft released two advisories that point to the use of wildcards in setting up phishing campaigns [62, 69]. Wildcards may be attractive to miscreants because they allow mapping multiple host names in their campaigns to the same IP address, for example. This can be useful in evading host name based blacklists, which rarely contain all host names belonging to a fraud campaign, with minimal effort.

Our primary goal in this chapter is to survey wildcard usage among benign, malicious, and neutral domains in the Internet. Within malicious uses, we consider scam, phishing, and malware web sites. Toward this goal, we query approximately 8 million domains for wildcard entries in the four most popular DNS record types. Since wildcards have legitimate benign uses, we also examine possibilities for distinguishing malicious uses of wildcards from their benign uses. The ability to do so may be helpful in identifying and effectively blacklisting malicious domains.

Working towards these goals, we arrive at the following key results:

- Prevalence: We find that a surprisingly large percentage of Internet domains use wildcards.

Specifically, 25-75% of domains in various data sets use wildcards, making this a much more popular DNS feature than we expected.

- Type: An overwhelming majority of domains using wildcards use them in their `A` records, which map arbitrary host names to IP addresses.
- Uses: Prominent users of wildcards include domain-parking businesses that wish to monetize unregistered domains and subdomains, web-hosting companies, and blogging and social-networking sites.
- Malicious sites: Malicious sites also make extensive use of wildcards, with spammers leading the pack with 75% of the scam-related domains in our data wildcarded.

5.2 Background

Wildcard records provide a simple method to map all host names within a domain or subdomain to a single DNS entry, as opposed to the one to one mapping provided by a non-wildcarded DNS record, or the few to one or one to few mappings that can be achieved through multiple records containing either the same name or the same data. Many use cases for wildcards could be dealt with without them by having many records all pointing to the same data. However, such records would be difficult to maintain, so wildcards provide increased convenience.

Wildcards in DNS were first defined in RFCs 1034 [63]. Later, RFC 4592 [50] updated and clarified the specification, providing more details and examples of intended behavior, and the interactions of wildcards with specific record types. A wildcard record is a DNS record of any type with a minor change to the name portion of the record. In a wildcarded DNS record, instead of the name being an exact host name, its least significant (leftmost) label in the name consists of a single asterisk character, as shown in Figure 5.1(a). Conceptually, the asterisk matches one or more labels at the left end of the DNS name. In this example, the `*.foo.com` is being used in place of `mail.foo.com` and `ns1.foo.com`. When a DNS query is made for `mail.foo.com`, seeing no match, the server will return results for `*.foo.com`, substituting `mail` for the `*`. Specific records override

www.foo.com	A	129.79.245.53
foo.com	MX	mail.foo.com
foo.com	NS	ns1.foo.com
mail.foo.com	A	129.79.247.191
ns1.foo.com	A	129.79.247.191

(a) without wildcards

www.foo.com	A	129.79.245.53
foo.com	MX	mail.foo.com
foo.com	NS	ns1.foo.com
*.foo.com	A	129.79.247.191

(b) with wildcards

Figure 5.1: Example of DNS provisioning of a domain with and without wildcards

the wildcard records. Since the record for `www.foo.com` is still present, the wildcard would not be considered when responding to a query for this host name.

The client receiving a DNS response cannot directly tell if the response was generated from a wildcard record or not; their use is transparent to the client systems. If a query for host name `name.foo.com` were matched from the wildcard record `*.foo.com`, the name on the record returned in the response will still be `name.foo.com` instead of `*.foo.com` as it is stored on the DNS server. We can however still tell if a wildcard is in use by directly querying for the wildcard name, in this case, `*.foo.com`. Since the wildcard record is the only one that would match such a query, if a response is given to such a query, it would let us know a wildcard record is present. Note that wildcard matches only work on one direction. Although the query for `*.foo.com` looks like the client has a wildcard in the query, it will only match an explicit wildcard record, not an arbitrary name in `foo.com` on the server.

5.3 Data Sets

Our goal is to study DNS wildcard usage in three contexts: domains judged as worthwhile or useful by Internet users, which we consider to be benign; domains from several blacklists, which we consider to be malicious; and a large general collection of domains, including both good and bad domains. Table 5.1 shows an overview of the data sets.

The first context in which we study wildcards is the domains determined to be useful by users in the Internet. For this purpose we use the `DMOZ` dataset described in Chapter 3. We assume that those links submitted and approved are those someone has judged to be worthwhile, and are

	DMOZ	Zone_Files	PHISHING	MALWARE	SPAM
Start Date	Sept. 17	Sept. 27	Sept. 22	Sept. 22	Sept. 22
End Date	N/A	N/A	Oct. 21	Oct. 21	Oct. 21
Frequency	Once	Once	Daily	Daily	Daily
Hosts	3,038,928	N/A	16,496	18,570	N/A
Domains	2,737,326	5,536,475	10,575	12,854	548,041
TLDs	3,235	7	306	259	327

Table 5.1: Overview of data sets used for investigating DNS wildcards

therefore unlikely to be malicious. We consider 2.7 million domains contained in this data set on September 17th, 2009.

The next context in which we study wildcards is domains known to be associated with malicious activity. For this context, we use host names extracted from the APWG and PhishTank phishing feeds, eSoft, MalwarePatrol, and CleanMX malware feeds, and SURBL spam feed, all described in detail in Chapter 3. We examine each of these feeds every day for a period of 30 days, extracting a total of 571,470 domains that were alive at the time of our receiving the feed. We refer to these data sets respectively as PHISHING, MALWARE, and SPAM throughout this chapter.

The last context we consider is a large general list of domains on the Internet. We obtain this list through the use of the seven generic top level domain zone files in our Zone_Files data set, described in more detail in Chapter 3. Specifically we use zone files from .asia, .biz, .com, .info, .mobi, .net, and .org. There were 110,728,143 domains contained in these TLDs, on September 27th, 2009, 58% of the total 192 million domains in the Internet at the time [108]. From these, we randomly sample at a rate of 5%, or 5,536,475, which we examine in this chapter.

5.4 Wildcard Prevalence

Wildcards can occur at all levels of the DNS hierarchy. We concentrate first on the domain level. We also briefly examine wildcards at the subdomain level.

	DMOZ	Zone_Files	PHISHING	MALWARE	SPAM
Domains					
Checked	2,737,326	5,536,475	10,575	12,854	548,041
Active	2,717,186	4,861,053	9,044	11,312	226,060
Inactive (%)	0.73%	12.2%	14.5%	12%	58.7%
Wildcards					
total %	24.52%	45.15%	32.09%	31.39%	75.10%
% A	18.76%	42.72%	27.79%	26.59%	72.30%
% NS	0.32%	5.53%	0.20%	0.19%	1.60%
% MX	5.72%	6.44%	4.10%	6.14%	6.83%
% CNAME	3.40%	3.75%	3.37%	4.49%	2.34%

Table 5.2: % of active domains with wildcards of each record type in each data set

5.4.1 Wildcards at the Domain Level

We look for wildcards in four DNS record types: A, NS, MX, and CNAME. From the entries in each data set, we determine the domain name part of each host name using the Public Suffix List [67]. For all domain names in these data sets, for example, foo.com, we query for *.foo.com for the four record types. All queries were run once for each domain in the DMOZ and Zone_Files data sets, but daily for the others that changed often in real-time. We also query for the NS record for each domain to ensure that the domain exists at the time of the query.

A large fraction of domains we surveyed used wildcards. Table 5.2 presents an overview of the number and types of wildcards present in each data set at the domain level. Between 1/4 and 3/4 of domains use wildcards, with the DMOZ data set showing the least prevalence of wildcards and the SPAM data set showing the most. Not only is the A wildcard overwhelmingly popular, its usage mimics general wildcard usage trends. Some domains have more than one type of wildcard, causing the percentages in the last four rows of Table 5.2 to exceed the total percentage of domains using wildcards.

5.4.2 Wildcards at the Subdomain Level

We now examine if wildcards are commonly used for subdomains as well. A domain can have multiple levels of subdomains. We can only infer wildcard usage for subdomains contained in host names present in our data sets. For example, if a data set contains the host name a.b.c.example.com

	DMOZ	PHISHING	MALWARE
Domain wildcarded	0.3%	4.0%	2.3%
No domain wildcard	0.1%	1.3%	0.6%

Table 5.3: Percent of wildcarded and non-wildcarded domains containing wildcarded subdomains

we could query for wildcard entries `*.a.b.c.example.com`, `*.b.c.example.com`, `*.c.example.com`, and `*.example.com`. We look for subdomain wildcards in the three data sets where host information is available: `DMOZ`, `PHISHING`, and `MALWARE`.

Table 5.3 shows the results of the subdomain level wildcard analysis. The first row of the table shows the number of wildcarded domains that had subdomain wildcards in the three data sets. The second shows the same information for non-wildcarded domains. Malicious domains are more likely than benign ones to have subdomain wildcards. However, overall there are not many wildcarded subdomains in our data. Even when domains have subdomain wildcards, they do not apply this to all of their subdomains. On average, counting only domains with any subdomain wildcards, domains in `PHISHING`, `MALWARE`, and `DMOZ` have 68.5%, 46.9%, and 33.7% of their subdomains wildcarded respectively. Because of the low number of domains containing subdomain wildcards, we focus only on domain level wildcards for the rest of this chapter.

5.4.3 Overridden Wildcards

Some wildcards may be overridden by specific entries. Exact matches for a query override wildcard matches. When a DNS query would have both an exact match and match a wildcard as well, the response is only constructed based on the exact match. The wildcard is in this case not considered at all. For example, a domain `foo.com`, may have a wildcard entry for `*.foo.com`, and a more specific entry for host name `a.b.foo.com`. This allows the domain to point `a.b.foo.com` to a different value than any other host name matching `*.foo.com`. Now that we have seen how often wildcards are occurring, an important consideration is if they are overridden by a more specific DNS entry.

Toward the goal of identifying overrides, we proceed as follows. For the `DMOZ`, `PHISHING`, and `MALWARE` data sets where we have host names in the feeds, we query the DNS for `A` and `CNAME` records corresponding to the host names and check if the results of this lookup match the results of

	DMOZ	Zone_Files	PHISHING	MALWARE	SPAM
A	10.7%	31.6%	19.0%	19.9%	6.7%
CNAME	17.4%	8.8%	17.5%	30.0%	2.8%

Table 5.4: Percentage of A and CNAME wildcards being overridden by specific entries

the wildcard lookup. If they are not the same answers, we consider the exact match to be overriding the wildcard. If we have multiple host names for one wildcard, we count it as an override if any of them do not match the wildcard entry. For Zone_Files and SPAM, we do not have exact host names, so we simply prepend www to the domain name. Though we do not know for sure that the host name so generated is in use, it is commonly used for web servers and may catch some overrides.

Notice that since our data sets are for web servers only, they do not contain name servers or mail servers. As a result, we cannot establish the presence of overrides for MX and NS wildcards by querying each domain for MX and NS wildcards and comparing the result to host names in the feed. This limitation is not severe since as seen in Table 5.2 MX and especially NS wildcards are not widely used.

Table 5.4 shows the percentage of A and CNAME wildcards being overridden in each data set. Wildcards are overridden in 2.8-31.6% of cases. The SPAM data set sees the least overrides. Some data sets witness overrides for CNAME wildcards more often than those for A wildcards and vice versa. The difference is most striking for the Zone_Files data set. Examining the overrides in this data set closely, we find that 25.4% (557,949) of wildcards in the Zone_Files data set are hosted on name servers in domaincontrol.com. Of these, 99.7% are A wildcards being overridden by a specific CNAME record. These account for 88.9% of the overrides of A wildcards in this data set. If we ignore wildcard entries on this name server, only 6.6% of remaining A wildcards in this data set are overridden, much closer the percentage of overridden CNAME wildcards in this data set. We conclude that wildcards are not frequently overridden in most data sets.

5.5 Wildcard Usage

In the previous section, we examined the prevalence of wildcards. Now we investigate their specific uses by the good, bad, and neutral. To group related wildcarded domains, we considered several

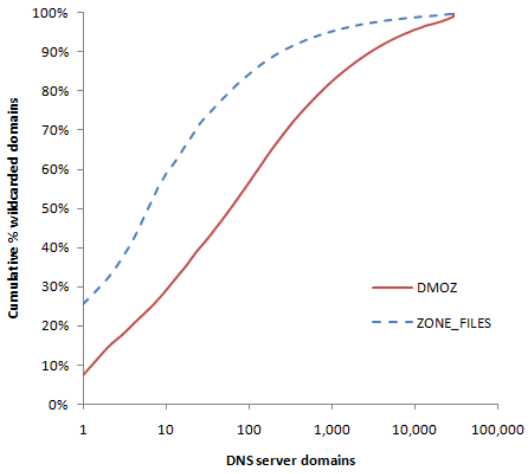


Figure 5.2: CDF of wildcarded domains served by each DNS server domain

options such as by Top Level Domain (TLD) and by Autonomous System (AS), and found it best to aggregate them by the DNS servers serving them. This grouping most intuitive out of those we considered because providers of DNS services, for example hosting companies, often provide a default configuration which most domains may choose. Similarly, large organizations running many of their own domains are likely to use similarly-provisioned servers. In fact, we aggregate even more by grouping wildcarded domains in terms of the domain of the DNS server.

5.5.1 Wildcard Usage Among Good Domains

The first data set we analyze is `DMOZ`, our set of good domains from a user edited directory. From this data set, we saw a total of 666,334 domains (24.5%) using wildcards. These were served by DNS servers belonging to 28,883 domains. Figure 5.2 shows a CDF of the wildcarded domains and the corresponding DNS server domains for this and `Zone_Files` data sets. We discuss the latter in Section 5.5.2. A key observation from this Figure is that just a few DNS servers are responsible for a disproportionate number of wildcarded domains. Specifically, 29.1% of domains in the `DMOZ` data set are served by just top ten DNS server domains.

We now consider the top ten DNS server domains serving the most wildcarded domains. Table 5.5 shows the total domains and wildcarded domains served by each. In looking over the domains

	Domains served	Wildcarded domains
worldnic.com	55,947	48,484
rzone.de	47,913	47,771
yahoo.com	23,194	21,835
namespace4you.de	17,611	17,409
kasserver.com	13,313	13,227
name-services.com	17,529	10,595
b-one.nu	9,471	9,406
ipower.com	9,058	9,057
register.com	13,853	8,077
mediatemple.net	7,869	7,705

Table 5.5: Top 10 DNS server domains serving the most wildcarded domains in the `DMOZ` data set

accounting for most wildcard usage, we find that all are operated by registrars or web-hosting providers. Both these entities tend to provide a default configuration to users which includes a wildcard record. This setup is useful because it reduces the support the registrar has to provide to a novice user. The user does not need to configure a DNS record for the host name he wants to use because they will all be taken care of by the wildcard. Even users who override these with specific records for individual hosts may choose to keep the wildcard record.

5.5.2 Wildcard Usage Among a General Collection of Domains

We now change our focus to the `Zone_Files` data set, a large collection of domains taken from several TLD zone files. In this data set, we saw 2,194,565 domains using wildcards (45.2%). These domains are also served by a small number of DNS server domains, only 32,644. Overall, we find that wildcarded domains are even more concentrated at a few name server domains than in the `DMOZ` data set; 58.9% of wildcarded domain in this set are served by name servers in 10 domains. Table 5.6 shows the top ten name server domains serving the most wildcarded domains in the `Zone_Files` data set. Four of the domains listed are in common with those in Table 5.5.

100% of the domains served by `sedoparking.com` and `dsredirection.com` are wildcarded. These two, along with the two others that have the highest percentage of wildcarded domains, `fabulous.com` and `parked.com`, belong to companies involved in domain parking: the domains server no actual useful content, just a template page filled with ads redirecting the user to other

	Domains served	Wildcarded domains
domaincontrol.com	1,138,877	557,949
name-services.com	179,130	147,697
worldnic.com	137,696	116,759
sedoparking.com	96,790	96,789
dsredirection.com	91,796	91,796
yahoo.com	82,747	80,669
register.com	72,827	62,137
secureserver.net	62,672	60,063
fabulous.com	39,166	39,137
parked.com	37,529	37,522

Table 5.6: Top 10 DNS server domains serving the most wildcarded domains in the `Zone_Files` data set

pages, mainly for the purpose of monetizing chance-visitors to the domain. Wildcards are very useful for parked domains. By directing visitors to a parking page, they allow monetization of all possible subdomains of a domain. However, not all parked domains are wildcarded. At least one provider of parking services we know of serves over 700,000 domains but uses wildcards on below 1% of them. The other major user of wildcards in this data set are web-hosting providers, as we saw in `DMOZ`. In fact, four of these are the same ones we saw in the top 10 from the `DMOZ` data set.

5.5.3 Wildcard Usage Among Bad Domains

Next, we look for wildcard usage in bad data sets, `PHISHING`, `MALWARE`, and `SPAM`. As we saw in Table 5.2, 32.1% of active phishing domains, 31.4% of active malware hosting domains, and 75.1% of active spam domains were using wildcards. Spam senders are using a far greater proportion of wildcards than anyone else, although all of these are using wildcards in significantly higher proportions than we saw for the good domains in `DMOZ`, which had 24.5%. Clearly the miscreants have figured out the flexibility offered by wildcards. For each domain used in their phishing, scam, and malware campaigns, they can simply swap a blacklisted host name with a new one without doing any extra maintenance of their DNS entries. This helps them defeat blacklists, which are often based on exact host names instead of domain names because the latter can incur false positives by penalizing good host names within a domain serving malicious campaigns. The proportion of `PHISHING` and `MALWARE` domains we see with wildcards is lower than what we saw in `Zone_Files` (45.2%). This is

	Domains served	Wildcarded domains
ixwebhosting.com	151	151
nshost.com.ve	139	139
rzone.de	63	60
yahoo.com	98	55
name-services.com	54	47
hosteurope.com	100	44
worldnic.com	48	42
hrnoc.net	33	32
register.com	32	30
namebay.com	128	29

Table 5.7: Top 10 DNS server domains serving the most wildcarded domains in the PHISHING data set

	Domains served	Wildcarded domains
freeservers.com	203	203
ixwebhosting.com	93	92
ipower.com	83	83
name-services.com	101	81
northsky.com	73	73
everydns.net	173	67
yahoo.com	63	59
servage.net	58	57
sorpresor.com	51	51
sitelutions.com	54	49

Table 5.8: Top 10 DNS server domains serving the most wildcarded domains in the MALWARE data set

likely because of the prevalence of parked domains in Zone_Files, who seem to be a major user of wildcards.

The top ten DNS server domains serving wildcarded domains in PHISHING, MALWARE, and SPAM, are shown in Tables 5.7, 5.8, and 5.9. They account for 21.67%, 22.95%, and 21.82% of the wildcard domains in these data sets respectively. This indicates a slightly lower concentration on the top name servers than we saw in the DMOZ data set, and much lower than we saw in the Zone_Files data set.

Another key observation from these tables is that many of the top-10 domains serving wildcarded domains are shared across all data sets. One reason for this commonality is domain parking. Some of the name server domains from the SPAM data set are associated with domain parking, and are

	Domains served	Wildcarded domains
name-services.com	16,699	14,764
tutby.com	6,167	5,966
domainservice.com	4,640	4,555
domainsite.com	3,202	3,200
domaincontrol.com	6,278	2,045
dsredirection.com	1,778	1,777
sedoparking.com	1,323	1,323
netstandardconsulting.com	1,296	1,296
peak-communications.net	1,180	1,180
dzcamera.net	941	940

Table 5.9: Top 10 DNS server domains serving the most wildcarded domains in the SPAM data set

probably there due to spam domains that have been taken down but still appear in our data set. These are fewer than 5% of the wildcards in SPAM so are certainly not the primary reason it has a higher proportion of wildcards than the others. Others are present because they are hosting providers. The most prominent example of this is name-services.com, which appears in the top ten from every data set. This and the few others from the three malicious data sets that are also top users in the other data sets may be large providers of malicious wildcards just because they are large providers who use wildcards by default and miscreants happen to use them. However, a majority that are the top users in these three data sets are not among the top users in the other two, making it likely that the miscreants are configuring wildcards intentionally.

Churn of Hosts Among Bad Wildcarded Domains

Miscreants can exploit the flexibility of wildcards to their advantage by simply swapping a blacklisted host name with a new one without having to change DNS entries. This can be a useful in evading blacklists, which are based on exact host names today. We now attempt to determine if such is the case. To do so, we examine if new host names matching an existing wildcard entry are being added to our feed of bad data sets over time. In this analysis, we focus on the PHISHING and MALWARE data sets, since the SPAM data set only includes domains, not host names.

Toward this goal, we calculate the daily churn of host names for each wildcarded domain in PHISHING and MALWARE data sets. For this, we compare the host names for a domain with those

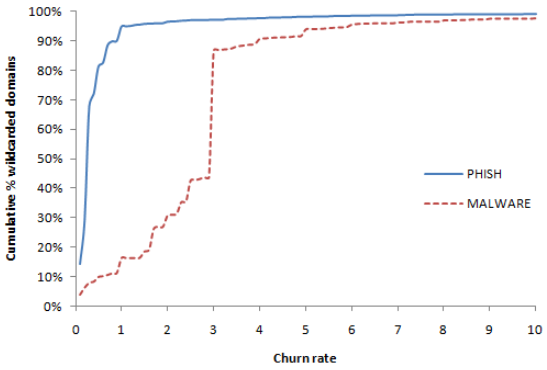


Figure 5.3: CDF of churn rate of malicious domains over 30 days

listed the previous day. The sum of the additions and deletions is the churn rate for the day. We average this over all days the domain is alive. We do not count the initial set-up or take-down of the domain since some domains may have existed before or continued to exist after our data collection. Domains only seen for one day are also not counted since there is no second day to compare to to derive a churn. Figure 5.3 depicts the CDF of churn over a period of 30 days.

For the `PHISHING` data set, the average churn rate is 0.64, a little more than one change every 2 days, and the maximum is 52. For malware, the average is 2.87 with a maximum of 32.5. Clearly, these numbers indicate that miscreants whose domains are active for more than a day, especially those serving malware, are taking advantage of the wildcard records to use new host names over time.

5.6 Identifying Malicious Wildcard Usage

Thus far, we have seen that wildcards are in wide-spread use among all types of domains in the Internet. Even though some types of bad domains use wildcards more commonly than good or neutral domains, there are no clear trends that would distinguish wildcard usage among such domains from others. The primary reason for this is that the largest wildcard users are domain registrars and web-hosting providers and many of them are common across all data sets. This is somewhat unsurprising, given that a recent report examining phishing attacks from the first half of 2009 [89],

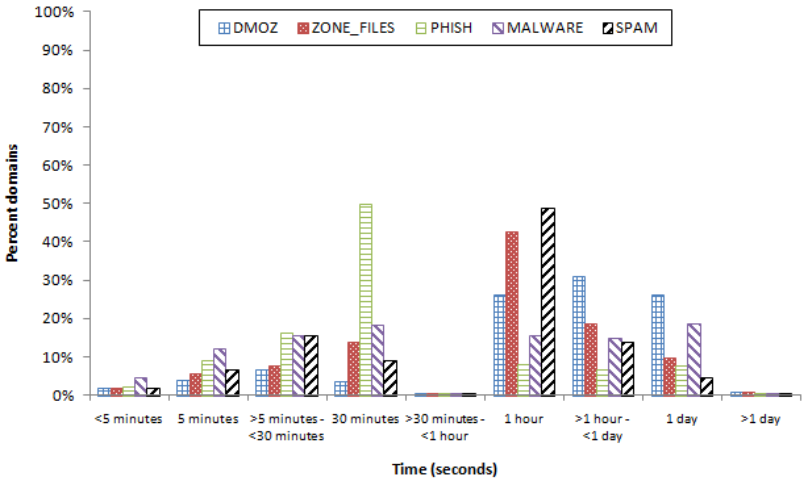


Figure 5.4: TTLs for `A` records in each data set

found that only 14.5% of domains used in phishing were actually registered by the phishers, the remaining were compromised domains that could belong to a known service provider.

In this section, we examine three additional features to help disambiguate wildcard usage by bad domains from other types of domains. First, we examine Time to Live (TTL) values on wildcard DNS records. Then we examine ASes corresponding to the wildcarded domains. Finally, we use the Google search engine to determine how many host names it knows of matching each wildcard, and to discover new hosts matching known wildcards.

5.6.1 TTLs of Wildcarded Records

To distinguish malicious uses of wildcards from good ones, we first examine the TTLs for each type of wildcarded records, and compare them across the three data sets. We focus on `A` wildcards, since they were the most common type. A histogram of TTLs for `A` records is shown in Figure 5.4.

A few TTLs are most popular: 5 minutes, 30 minutes, 1 hour, and 1 day. The most significant difference we see between data sets is `PHISHING` has a large spike at 30 minutes. In general, we find that wildcards in the `PHISHING`, `MALWARE`, and `SPAM` data sets have shorter TTLs than those in good and neutral data sets, with 30 minutes and 1 hour being most popular values for bad wildcarded domains. This is intuitive because shorter TTLs allow miscreants to quickly update the IP addresses

corresponding to malicious host names. Given the benefit to miscreants of strategies that rely on short TTLs such as fast flux, described in Chapter 4, examining TTLs corresponding to wildcarded records appears to be a promising avenue for investigation.

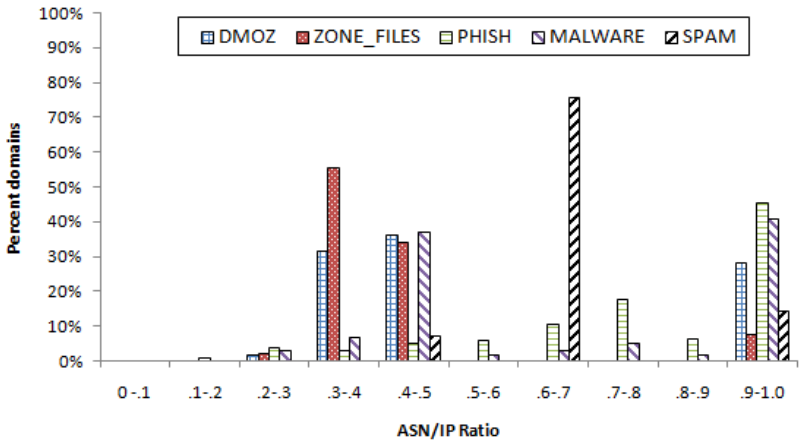
5.6.2 Autonomous Systems Pointed to by Wildcards

Many of the malicious domains are hosted on bots in geographically diverse Internet locations. ASes are one way to measure this diversity. The second feature we examine to distinguish malicious from good wildcards is how often are IP addresses corresponding to wildcard records are spread over multiple ASes. This is straightforward to do for `A` wildcards, since the right hand side of these records directly provides an IP address. For `CNAME`, `MX`, and `NS` records, which point to a host name instead of an IP address, we simply resolve the hosts on the right hand side to IP addresses. For all wildcard types, we see some difference in the results across various data sets, however, we focus on `A` and `CNAME` wildcards in this discussion since these show the greatest difference, enough that they are useful in detecting many cases of malicious wildcard usage.

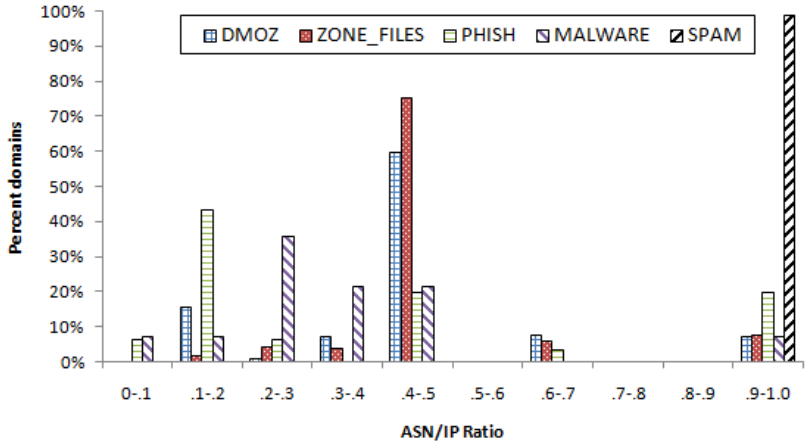
A histogram of the ratio of Autonomous System Numbers (ASNs) to IP addresses for wildcarded `A` records is shown in Figure 5.5(a). The most notable observation here is that a majority of `SPAM` wildcard domains with multiple IP addresses have a ratio of ASNs to IP addresses between 0.6 and 0.7. Very few of the good data sets are in this range. In fact, `PHISHING` and `MALWARE A` wildcards are much more likely than `Zone_Files` and somewhat more likely than `DMOZ` to be in the 0.9 to 1.0 range.

Figure 5.5(b) shows the ASN/IP ratio for wildcarded `CNAME` records. Here, the `SPAM` data set almost all ends up in the 0.9-1.0 range, while fewer than 10% of the wildcarded `CNAME` records from the good data sets do so. Phishing and malware sites are significantly more likely than good ones to fall into the ranges from 0.1 to 0.4.

Overall, this method looks like a good one for identifying wildcards associated with spam sites, and a decent one for wildcards associated with phishing and malware sites. The only issue with it is that it relies on the wildcard entry pointing to multiple IP addresses, since otherwise, the notion of geographical diversity makes no sense. This happens for 1.6 - 4.2% of domains with `CNAME` wildcards



(a) A



(b) CNAME

Figure 5.5: Ratio of number of ASNs associated with each wildcard A and CNAME record to number of IP addresses pointed to by record

and 0.5 - 27.2% of A wildcards depending on the data set. In the SPAM data set, it happens for 18.2% of A wildcards and 41.2% of CNAME wildcards. This data set is also the one where the ratio is most different from the good data sets, indicating that it would be effective a significant amount of the time for identifying wildcards associated with spam.

	DMOZ	Zone_Files	PHISHING	MALWARE	SPAM
Domains checked	6,717	9,867	1825	2321	4,057
Domains responding	6,587	4,596	1089	1263	475
% indexed	98.1%	46.6%	59.7%	54.4%	11.7%

Table 5.10: Wildcarded domains from each data set queried at Google

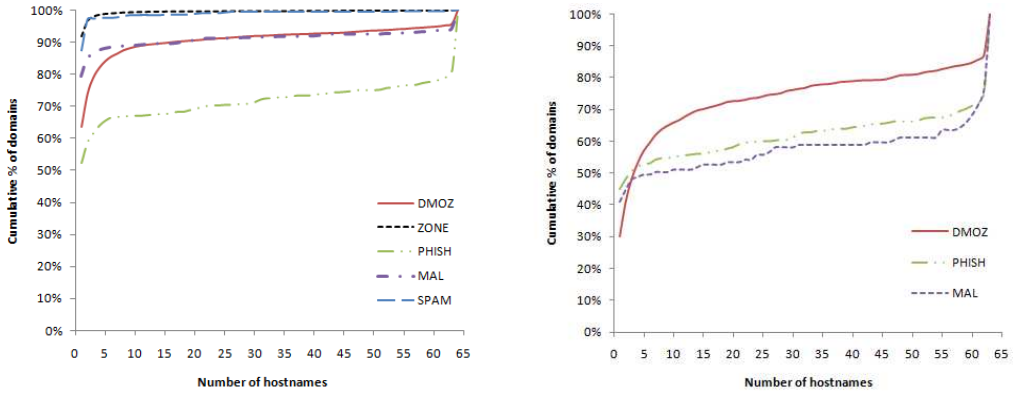
5.6.3 Host Names Represented by Wildcards

Technically, a wildcard entry in the DNS can match any host name. However, in practice, a site may only use some of these host names. The number of host names in use matching each wildcard is the third feature we examine to distinguish malicious from good wildcard uses.

Blogging and social networking sites often provide a subdomain for each user. Out of 170 such sites we investigated, 52 support subdomains for each user. Of these 52, all do so using wildcard entries, 37 of them with `A` wildcards, and the rest with `CNAME`. As a specific example of this, `Windows Live Spaces` provides a subdomain for each user, all handled by a single wildcard entry, and claims 175 million users [61]. Even the smallest blog site we have found using wildcards supports over 10,000 subdomains.

We now investigate if Google searches can reveal new host names covered by our wildcards. Toward this goal, we queried the Google search API [23] for a sampling of the domains with `A` or `CNAME` wildcards from each data set, using site restriction to make sure all responses were from the domain we were interested in, not external pages with the domain in their text. This gives us an idea of how the wildcard is being used, subject to a maximum of 64 results imposed by the Google API. Table 5.10 shows how many domains were queried from each data set, and what percentage were found in the Google index.

We find that a large percentage of domains we queried were indexed by Google. Over half from `Zone_Files` were not indexed, probably due to the large amount of sites devoid of useful content, such as parking pages. On the other hand, over half of the pages in two of our sets of malicious Uniform Resource Locators (URLs), `PHISHING` and `MALWARE` were indexed. From `SPAM`, a large majority were not indexed by Google. This is perhaps because the URLs associated with them are only linked through email, so the Google crawler would have never seen them. Also note that not being indexed



(a) All Host Names

(b) New Host Names Only

Figure 5.6: Cumulative percent of wildcarded domains in each data set with the given number of host names found in the Google index

does not necessarily mean Google has not crawled the page. It is possible for Google to intentionally exclude pages with known malicious content. Since so many of the malicious domains are found by Google, the presence of a domain in the Google index can not be used to reliably determine if a site is malicious or not.

Out of the domains that did return Google results we examine how many results were returned. Results are shown in Figure 5.6(a). The most notable result here is that wildcards from `PHISHING` correspond to a higher number of hosts known by Google than wildcards from other data sets. For the other data sets, meaningful distinctions are hard to make, since `SPAM` and `Zone_Files` results are similar to each other, as are `MALWARE` and `DMOZ`. While it can not be said with certainty that wildcards representing large numbers of host names are associated with phishing, it is certainly an indication that further scrutiny is required to see if they are phishing sites. While a client could not directly determine the number of hosts a wildcard represents, any organization who crawls the Web should be able to provide data on how many host names they have seen in a domain name, making this check practical.

Figure 5.6(b) shows how many host names Google returned that were not found in our data sets. Here, only those data feeds that contained host names are considered since no conclusions can be drawn from data sets containing only domain names. For most domains Google indexed

in the PHISHING and MALWARE data sets, it knows of several host names not in our data set. This indicates that blacklisting could be improved by directly including wildcard entries instead of exact host names. Additionally, there is significant difference between the PHISHING and MALWARE data sets where Google knows of at least 10 new host names for 45% and 49% of domain names, and DMOZ, while it knows of 10 new host names for only 34% of the domain names. This indicates that DMOZ, whose role is only to index sites people judge as worthwhile, actually is more thorough in its coverage than the blacklists, who need to be thorough to be completely effective. This may be due to the much quicker changes in malicious activity as compared to good sites, whose names do not tend to change.

5.7 Discussion

It is clear from the results in this chapter that miscreants are taking advantage of wildcards. However, we also saw more benign uses than we had expected. Among the malicious, spammers are using them most, and are the least likely to override them. This enables them to use the wildcards effectively to evade blacklists by switching to new host names with little effort. There is also a significant churn among host names matching wildcards in malware and phishing domains, suggesting that they too are likely taking advantage of the wildcards to escape exact host-name-based blacklists.

Because of the mix of good and bad uses, the fact that a wildcard is in use is unlikely to be a good feature on its own. However, along with other characteristics, wildcard usage may be used to flag malicious web sites proactively. Specifically, we investigated three features in Section 5.6: short TTLs, distinct ratios of IP addresses to ASNs when the wildcard pointed to multiple IP addresses, and the number of hosts in the domain known to Google.

Determining if a DNS response is coming from a wildcard is a low overhead operation that could be done by software on the end user's computer without introducing additional delay, or done by a monitoring system just as efficiently. The monitoring system could make a query for the record matching the wildcard record for each domain under consideration, as we do in this study. At an end user system, this can be done in parallel with the normal DNS query for the domain, and the results compared when both responses are received. TTL information is contained in the DNS

response, so it also adds no additional overhead. Determining the ratio of ASes to IP addresses, or how many hosts are known in a domain are higher overhead operations, although the former can be done efficiently through longest prefix matching and the latter might be made efficient by a service built to respond to such queries.

Miscreants could attempt to evade detection based on wildcards and still use a large number of host names by creating separate DNS entries for each. However, their inherent constraint to move across a large number of host names can be detected. The monitoring system could keep track of how many host names within a domain it has seen resolving to the same IP address, and flag domains passing a certain threshold of host names.

6

Orphan DNS servers

6.1 Introduction

Domain Name System (DNS) records stored at the Top Level Domain (TLD) DNS servers are commonly of two types: those that contain the host name of the domain's name server (associating `example.com` with `ns1.example.com`), and those that contain its IP address (associating `ns1.example.com` with an address). This chapter focuses on a special case of IP addresses records known to the TLD DNS servers: those where an IP address record exists for a DNS server whose parent domain has ceased to exist. An example of such a case is if there were an IP address record for `ns1.example.com` at the `.com` TLD when `example.com` itself was non-existent. Servers such as `ns1.example.com` have come to be referred to as *orphan name servers* in the operational community.

Orphan name servers have been found in use by malicious web sites [82]. Orphan servers may be attractive to miscreants for a few reasons. For example, if they are using them for purposes other than DNS servers, having records for them on the TLD DNS server means the miscreants may not even have to run their own DNS server. Orphans related to malicious activity may also naturally arise from enforcement efforts by Internet Service Providers (ISPs) removing `NS` records of domains found to be malicious. The DNS server may already have been in use by other domains belonging to the same miscreant, and may continue to be used as such if the `A` record is left.

We focus on studying the prevalence of orphan name servers in the Internet and to understand

what purposes, good and bad, they are used for. Within malicious uses, we consider their use by phishing, scam, and malware web sites, as well as spammers. Significant benign uses obviously would make it more difficult for use of orphan servers to be used as a criteria to determine malicious intent. The questions we ask are:

- How prevalent are orphan DNS servers?
- How long do orphan name servers live?
- Are orphan name servers being used as anything other than as DNS servers?
- Are orphan name servers commonly used for malicious purposes?

We examine an average of 106 million domains (60% of the domains in the Internet at the time of our measurements) on a daily basis for a period of 31 days to investigate aspects of orphan DNS servers. We arrive at the following key observations:

- Prevalence: 1.7% of DNS servers we examine are orphans. Seemingly small, this amounts to 46,369 servers.
- Distribution: A surprisingly large proportion of orphans, over half, are in the `.info` TLD, although it contains two orders of magnitude fewer address records than the largest TLDs. 18.8% of servers in `.info` are orphans as compared to just 0.4% in the much larger `.com`.
- Lifetime: The median lifetime of an orphan server is 8-9 days. A few last over a month. This offers a significant window of opportunity to miscreants, especially given that finding existing orphans is not difficult.
- Use by miscreants: Cyber-criminals have discovered that orphans can be exploited. A significant portion of IP addresses associated with orphans, 4.4%, are known to send spam. About 1% of domain using orphans are listed on phishing and malware blacklists, although the orphans themselves are not often blacklisted. However, over a 26.1% of orphans serving a domain and 43.8% of the domains using them are indirectly associated with malicious activities.

fake.com	NS	ns1.fake.us
fake.com	NS	ns2.fake.us
example.com	NS	ns1.example.com
example.com	NS	ns2.example.com
ns1.example.com	A	1.2.3.4
ns2.example.com	A	1.2.3.5

Figure 6.1: Sample portion of a TLD zone file

6.2 Background

Here, we discuss the DNS records important to understanding orphans. We also present a scenario which may lead to the creation of an orphan.

6.2.1 DNS Background

The DNS server at each TLD maintains a *zone file* containing two types of records relevant for studying orphans. These are the `NS` (name server) and `A` (address) records for each domain falling within that TLD. The `SOA` record is present in all DNS zones, but not important in the discussion of orphans. Additional records may also be present for DNSSEC, but they are have no bearing on the study of orphan name servers.

The `NS` records provide a mapping from a domain to the names of its DNS servers. Taken together, they can also be used as a listing of all of the domains contained in the zone. Figure 6.1 shows a sample portion of a zone file for `.com` TLD. Here, two name servers each for domains `fake.com` and `example.com` are listed in the `NS` records. However, names do not suffice. Clients need the corresponding IP addresses to contact the name servers, but the authoritative source for this IP address is the server itself. Accordingly, `A` records corresponding to the `NS` records are also needed in the zone file. These records are referred to as *glue* records. Notice, however, that `A` records for `ns1.fake.us` and `ns2.fake.us` are not listed in Figure 6.1 because they they may be obtained from the name servers for the `fake.us` domain.

fake.com	NS	ns1.fake.us
fake.com	NS	ns2.fake.us
example.com	NS	ns1.example.com
example.com	NS	ns2.example.com
bad.com	NS	ns1.orphan.com
ns1.example.com	A	1.2.3.4
ns2.example.com	A	1.2.3.5
ns1.orphan.com	A	1.2.4.4

Figure 6.2: Sample portion of a TLD zone file containing an orphan name server record

6.2.2 Orphan Creation

We describe how orphans are created through a scenario. A domain, `orphan.com`, is created, leading to the addition of an `NS` record to the zone file: `orphan.com NS ns1.orphan.com`. The host `ns1.orphan.com` is in `orphan.com`, so to avoid circular dependencies, its address record, `ns1.orphan.com A 1.2.4.4`, is also added to the zone file. Other domains may also use this server, resulting in the addition of `NS` records for them as well. If `bad.com` uses this server, then `bad.com NS ns1.orphan.com` will be added.

Now imagine that some time later `orphan.com` is removed. This may occur for one of several reasons, such as because the domain was hosting a malicious activity, or just due to the registrant letting the domain expire. In this case, the above mentioned `NS` record for the domain is deleted from the zone. However, what happens of the `A` record? One option is to delete it. Another option is to let this record live on, to account for cases when other domains in `.com` or other TLDs depend on it, for example, the above mentioned `bad.com`. Deleting the `ns1.orphan.com A` record would harm `bad.com`, perhaps taking it completely offline if this is its only DNS server. Keeping this `A` record prevents the deletion of this single domain from affecting others which may rely on the same server. This leads to the presence of orphan DNS servers. The sample zone from Figure 6.1 after these additions and deletions is shown in Figure 6.2.

The above scenario is of course not the only way an orphan server may appear. Some may arise out of configuration errors, such a simple typo where the name associated with an `A` record is not what it should have been and so this incorrect name does not match any existing domain. Others may occur due to lax registrar policies. While glue records are the only reason an `A` record *should* exist in a TLD zone, mechanisms may not be in place to actually enforce this. Lack of enforcement

of such policy would allow such records to intentionally be added directly to the TLD zones.

6.3 Methodology and Data Overview

In this section we describe our method for locating orphan name servers and finding who is using them.

6.3.1 Finding Orphan DNS Servers

We use three different techniques for discovering orphan name servers in order to be exhaustive where we have the necessary data, and still find some orphans where we have less data available. These techniques are direct analysis of zone files, DNS queries based on zone file information, and DNS queries based on malicious feeds.

Analysis of zone files

Zone files for a TLD are the ideal way to find all orphans in that TLD. We use the zones files for 6 TLDs: `.asia`, `.com`, `.info`, `.mobi`, `.net`, and `.org` [2, 16, 68, 85, 109] from our `Zone_Files` data set. These TLD accounted for 60% of the 177 million domains on the Internet at the time of our experiments [106]. We obtain these zones through file transfers from the organizations that administer them, so they should be in a consistent state.

To find an orphan in a zone file, we use a simple algorithm. We first examine the entire list of `A` records in the zone file. For each host name listed in an `A` record, we extract the domain name. We then check if a `NS` record exists for the domain, to verify if the domain itself exists. If such a record does not exist, then we classify the name from the `A` record as an orphan name server. Returning to Figure 6.2, when we come across the `A` record `ns1.orphan.com A 1.2.4.4`, we then search the NS records in the zone file for any `NS` records for the domain `orphan.com`. Since no such records exist, we classify `ns1.orphan.com` as an orphan.

To locate which domains are using the orphan name server, we search the zone files for NS records that point to orphans. In this case, we would find bad.com.

DNS Queries from Zone Files

While our TLD data contains the largest TLDs, a notable deficiency is that it does not contain any country code Country Code Top Level Domains (ccTLDs) because operators of these TLD are not willing to share them even for research purposes. Finding orphan name servers exhaustively for these TLD is not possible. However, we can still find potential orphans through the zone files available to us.

Instead of looking at the A records, as we did earlier, we now look at the name servers pointed to by the NS records. If the name pointed to is in one of the TLDs we have a zone file for, we ignore it because if it were an orphan, it would have been identified through the previous method. If it is not, as is ns1.fake.us in Figure 6.2, we perform two DNS queries. First, we extract the domain name from the name server's host name (in this case fake.us), and perform an NS query on it. This query checks if the domain name the server is in exists. Orphans can only be present if the domain name does not exist. If this query fails, we perform an A query on the name server's host name (ns1.fake.us). If this query succeeds, the host name is an orphan. Otherwise, it is just a misconfiguration of a NS record pointing to a server that does not exist.

This method directly finds users of the orphans as well. The domain on the NS record used to find the orphan is a user of that orphan.

DNS Queries from Malicious Feeds

To get a glimpse of the use of orphans in hosting malicious web sites, including domains not seen by the above two methods, we bring in one more data source. We examine five different live feeds of phishing and malware-hosting sites, APWG, PhishTank, eSoft, CleanMX, and MalwarePatrol. We perform NS queries to identify their name servers, and then we perform DNS queries to check if these servers are orphans exactly the way we did in Section 6.3.1.

Start date of zone files	2009-04-01
Days of zone files used	31 Days
Domains represented in zone files	106 million
`A` records in zone files	2.2 million
`NS` records in zone files	249 million
Out of zone name servers looked up	241,179
Start date of malicious feeds	2009-04-16
Days of malicious feeds used	14 Days
Hosts in malicious feeds	242,752
Domains in malicious feeds	9,554
Name servers for domains in malicious feeds	48,042

Table 6.1: Overview of data used for orphan detection. Numbers presented are daily averages

Limitations of Data Collection

The above mechanisms allow us to find all orphan name servers contained in or used by domains in the DNS zones we can access. It also allows us to find ones used by domains in our malicious feeds. However, it is still not a complete picture of all orphans. To obtain a complete list is unfortunately not practical due to data limitations. Some orphans likely exist in zones we do not have access to. Some more may be used by domains not present in our feeds or zone files. The effect of this limitation is that we under-estimate the prevalence of orphan name servers. Similarly, while all of these mechanisms automatically locate users of orphans, the orphans may have more users not contained in the zone files or malicious feeds we have access to. This leads to an underestimation in the users of orphans as well.

6.3.2 Data Overview

We receive new zone files for each zone on a daily basis, and new malicious feeds every hour. We repeat the process described above each day on the new data. For this analysis we use 31 days of zone files and 14 days of the malicious feeds. Table 6.1 provides an overview of the data used to locate orphans.

We see that as expected, there are over double the number of `NS` records than there are domains, since most domains use multiple name servers. Also as expected, there are far fewer `A` records than domains, since many domains share name servers. A majority of the name server IP addresses were

contained in the zone files themselves. Only about 10% of them were out of the TLD zones and required a DNS look-up. The malicious feeds provided fewer servers to look up than the zone files, but are useful for providing more diversity in the potential orphan users.

6.4 Characteristics of Orphans

We now examine the prevalence, distribution, and lifetimes of the orphans we found.

6.4.1 Prevalence of Orphan DNS Servers

We begin by examining the prevalence of the orphans. As discussed in Section 6.3.1, due to data limitations, this should be considered as a lower bound on the prevalence of orphans. Overall, we find a total of 46,369 orphans. Each day, we record an average of 15,962 orphans. Many are present in our data for several days. 39,443 (85%) of orphans are in the TLDs we have zone files for, representing 1.7% of the `A` records in our TLDs, a small but significant percentage. For these, we know that they are all the orphans in these TLDs.

6.4.2 Distribution of Orphan DNS Servers

We now look at the distribution of orphans in the Internet in terms of the domains, TLDs and Autonomous Systems (ASes) they belong to, as well as their IP addresses.

Distribution in Domains and TLDs

Although we saw 46,369 orphans, they are likely not independent of each other. We can quantify relationship among them by looking at the number of domains that contained orphans. We find orphans in 23,153 domains, an average of 2.0 orphans per domain that has them.

Since TLD policies may affect the presence of orphan servers, we examine the number of orphans contained in each TLD for which we have zone files. These results are seen in the top part of

TLD	# orphans	A records	%
.info	26,111	139,126	18.8%
.mobi	433	4,062	10.7%
.asia	99	1,313	7.5%
.org	7,715	206,513	3.7%
.com	6,428	1,566,392	0.4%
.net	530	331,896	0.2%
.us	3,931	n/a	n/a
.cm	2,771	n/a	n/a
.cn	2,715	n/a	n/a
.br	2,495	n/a	n/a
.de	2,413	n/a	n/a
.ws	2,344	n/a	n/a
.kr	2,118	n/a	n/a
.ru	1,801	n/a	n/a
.ca	1,368	n/a	n/a
.in	1,340	n/a	n/a
.jp	1,182	n/a	n/a

Table 6.2: Orphan name servers by TLD, as compared to the total A records in each TLD zone

Table 6.2. Somewhat surprisingly, we see that .info has a much higher percentage of its A records as orphans than any other TLD. 26,111 (18.8%) of its 139,126 A records were orphans. In fact, .info contains four times as many orphans as the much larger .com which has an order of magnitude more A records. Notably, the two zones run by Verisign, .com and .net, have by far the least percentage of their A records as orphans as compared to the other TLDs. .info and .org are both operated by Afilias. This indicates that perhaps Verisign makes policy decisions that discourage orphans.

We also checked the TLD distribution of domains containing orphans, instead of the orphans themselves, to determine whether these high numbers were caused in part by many orphans in a few domains. This did not change the results significantly, including no change in the ordering of which TLDs had the most orphans.

Due to lack of zone files, we can not determine what percent of A records in other TLDs are orphans. We can also not determine the true number of orphans in other TLDs. However, even with the information we have, a few other TLDs stand out. Notably, 11 TLDs each have over 1,000 orphans, shown in the bottom part of Table 6.2, the most being 3,931 in .us. Since our methodology under-counts the number of orphans in these TLDs, the actual numbers are likely to be higher. In total, we find 123 TLDs containing orphans, although many of these contain only a few.

IP Address Distribution

As with orphans in the same domain, orphans pointing to the same IP address are also related since these are using the same physical machine. Collectively, we find 14,411 IP addresses in use by orphans, an average of 3.2 orphan names pointing to the same physical machine.

To determine if certain networks are responsible for disproportionate numbers of orphans, we translate these IP addresses into the corresponding AS numbers using the service offered by Team Cymru [103]. We find that 14,291 of these IP addresses belong to 1,960 ASes, an average of 7.3 IP addresses of orphans per AS that has them. This implies that about 6% of the approximately 30K ASes that originate routes on the Internet contain orphans. The remaining 120 are IP addresses with no corresponding AS, such as private IP addresses. These are in a sense doubly misconfigured, since they are orphans and point to addresses which are not publicly accessible.

Most ASes containing orphans only have a few, with 89% having 10 or fewer, and 43% having just one. In these cases, it is likely that the operators of the AS themselves are not responsible for the presence of these orphans, at least not intentionally. Instead, these are either the result of mistakes, or set up intentionally by some other party. However, we also see four ASes each containing over 100 IP addresses of orphans. In these few cases with many orphans pointing to the same AS, the operators may be responsible for the behavior, but there may be another explanation. These four top ASes belong to commercial web-hosting companies; it is possible that this behavior is instead due to their customers creating such records pointing to the hosted address. However, if this were the case we would expect to see the same behavior in many more ASes.

6.4.3 Lifetimes of Orphan DNS Servers

Next, we examine the lifetimes of the orphan name servers in our data. Each day we check if orphans recorded on the previous day are still present to examine how long each orphan persists. The results are shown in Figure 6.3.

We expected orphans to live for long periods of time, owing to the negligence in noticing them. In reality, many are shorter-lived. Specifically, 12% of them only live for a single day. These orphans

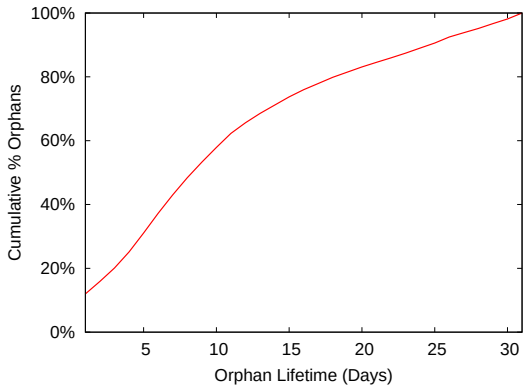


Figure 6.3: Orphan Lifetimes (CDF)

may have been created unintentionally and then removed once somebody noticed the error. Under 2% of them lasted for our entire 31-day data window. The median lifetime for an orphan is between 8 and 9 days, suggesting that many orphan name servers might exist for tasks that are completed in a short amount of time, such as hosting malicious web sites which are known to live only for a few days.

6.5 Uses of Orphans

In this section, we examine how orphan name servers are being used. We approach this question in a few ways. First, we examine their use in the most expected way, as a DNS server. Next, we examine the services running on each orphan. Then, we examine the domains with a large number of orphans to learn how they are using the orphans. Finally, we look specifically for malicious uses.

6.5.1 Use as DNS Servers

We now examine how many domains are actually using orphans as their DNS servers. In total, we see 212,850 domains using orphans as DNS servers, an average of 4.6 per orphan. Figure 6.4 shows that *almost 1/3rd of the orphans are actually in use as a name server by at least one domain.* We also find a few orphans being used by a large number of domains (not shown in the Figure). Specifically, two orphans are each used by 30K domains and two by 15K.

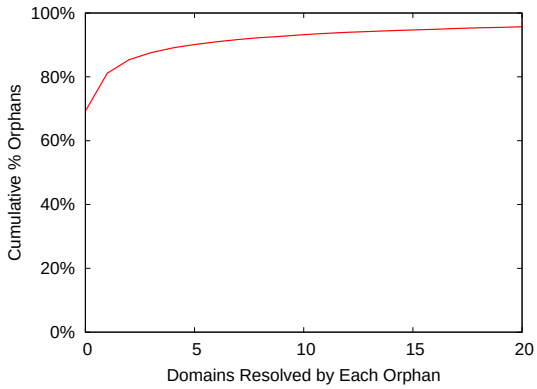


Figure 6.4: Number of domains using each orphan name server (CDF)

For the rest of the orphans, 69%, we were unable to find any users at all in our data. This is unexpected since the only purpose of `A` records in a TLD zone is to give addresses of name servers. This may suggest that these orphans are simply present as a result of oversights in keeping zone file contents up-to-date. However, perhaps they are used for other reasons. We investigate one such possibility in Section 6.5.2. Since those serving nobody can only have been found through zone file analysis, this also gives an idea of the percent in other TLDs we may be missing without access to such information.

6.5.2 Running Services

Several of the host names we have collected belonging to orphan name servers begin with `www`. Because `www` is a common label for hosts that run an `http` server, we check if port 80 is open on each orphan. If so, the orphan may be used as a web server. We also check if port 25 is open to see if they may be running an `smtp` server. 33% of the unique IP addresses of orphans appear to be running an `http` server, based on having an open port 80, and 22% appear to be running an `smtp` server, implying that they can potentially be used for sending e-mail.

From orphans with an open port 80, we select 1,000 which we do not see used as a name server. From these orphans we request a web page. We find that 42.9% of orphans return nothing despite having an open port. While we can not determine what content they are serving, we can not say they are serving nothing, since they may be using cloaking, ignoring our request since it does not

have a standard user agent. 27.9% of the orphans we checked display the Apache welcome page. Although they may be serving content on a URL somewhere other than the root of their domain, but there is none at the root of the domain. Although some web sites may be set up in this manner, we have no good way of guessing which path may be used in order to check it.

We classify the type of content on those which do return real content by keyword searches on the returned pages and manual inspection. 29.2% of orphans that had content at the root of their domain included advertising, blogs, and scam sites including the well known "Canadian Pharmacy" [97]. The Canadian Pharmacy scam was on 1.4% of the orphans we checked. Although these are not being used as DNS servers, this shows that miscreants may leverage orphans for other malicious purposes.

6.5.3 Domains with the Most Orphans

Some domains contain a very high number of orphans, the ten highest all contain more than 50, and the largest has 1,198. Although we can not determine for sure what all of them are doing, we do see one surprising activity.

A distinct behavior is seen among seven of the top ten domains containing the most orphans. The domain names of these tend to contain substrings like "delete" or "old". Such domains appear to contain name server records copied from domains that have or will soon expire. For example, we may see an orphan named `ns1.example.com.deleted.info`. Usually in this case, in addition to `deleted.info` not existing, the other domain implied in the host name, `example.com` does not exist either. However, if we send this server DNS queries related to `example.com`, it responds as if the domain existed, and it was a name server for the domain.Checking IP addresses, if we ask this server for the IP address of `ns1.example.com` we get an address matching that of `ns1.example.com.deleted.info` (itself). This server appears to be `ns1.example.com` simply with its host name changed.

Based on this behavior, we conjecture that registrars move name servers from expired domains to another domain as orphans to minimize disruption to other domains they administer. Other domains that use a server from the expired domain for name resolution can then have their `NS` record changed to point to the new orphan, perhaps automatically by the registrar, minimizing

Data Source	% of Orphan Users	# Domains
Google malware	0.4%	897
Google phishing	0.0%	4
Phishing Feeds	1.1%	2,250
Malware Feeds	1.3%	2,838
Scam Feed	0.0%	22

Table 6.3: Domains using orphans appearing in sources of malicious data

disruption to the service of these domains. Since these are orphan name servers, they have the same resolution behavior as before the original domain expired. If they were not orphans, then an additional query would be needed to look up the authoritative name server for the domain to which the name server was moved, creating an additional unintended dependency on the server for that domain.

6.5.4 Malicious Uses of Orphans

Given the attractive features of orphans for malicious activity, the observation that orphans are being used to host scam sites as we saw in Section 6.5.2, and the use of orphans by malicious sites found in [82], the connection between orphans and malicious activities deserves investigation. We search for malicious users of our orphan name servers in multiple sources described in Chapter 3. These are `SafeBrowsing` from Google, the `eSoft`, `CleanMX`, and `MalwarePatrol` lists of malware hosting sites, the `APWG` and `PhishTank` phishing lists, and the scam list from `SupportIntelligence`.

Table 6.3 shows how many malicious domains are using orphans as their DNS server. There are 212,850 total domains using orphans in our data. Of these users, 2,250 (1.1%), using a total of 86 orphan name servers, are in our phishing feeds. Similarly, 2,838 (1.3%), using a total of 53 orphan name servers are in our malware feeds. Other sources of malicious data also have orphans associated with them, as seen in the table. A small fraction of domains *containing* orphans appear in blacklists as well, the most significant being 76 appearing in the Google malware blacklist. Although miscreants are using orphans, orphans themselves are not often blacklisted.

We additionally look for the IP addresses of orphans in the `SBL` and `XBL` from Spamhaus. We find 4% of IP addresses of orphans appear in the `SBL` and 0.4% in the `XBL`. When we look only at

the 500 IP addresses hosting the most orphans, the SBL results are significantly different, 13.4% are blacklisted. We conclude that a significant fraction of orphans send spam.

Inferring Maliciousness

We find only a small number of orphans being used for malicious purposes directly. However, sources of malicious domains are by their nature incomplete. We now infer additional orphans which are potentially malicious.

The orphan name servers and the domains using them can be viewed as a bipartite graph. There are two sets of nodes in the graph, one representing the orphans, and the second the domains using them. An edge connects a node in the orphan set to a node in the domain set if and only if that domain uses that orphan as one of its name servers. We can construct our set of orphan nodes in two ways. We can identify them with unique host names for the orphans, or we can identify them with unique IP addresses for the orphans. The IP address based graph tends to be smaller since many orphans have several host names, often in different domains.

Suppose a domain node is located in a list of malicious domains. The orphan nodes that are adjacent to it are associated with malicious activity and should be implicated, since some business or social relationship exists between the operators of both. For the same reason, the orphans then implicate the domains adjacent to them in the graph. By repeating this process, every node in a connected sub-graph with a malicious domain gets implicated.

We identify each of the connected sub-graphs in the bipartite graph described above. If any node in the sub-graph is marked malicious, then every node in the sub-graph is implicated. We then count the number of orphans and domains implicated. We compute this based on two weeks of data. Since orphans serving nobody can not be implicated, they are not included in our graph construction and the implicated orphans percent is taken out of just the orphans who serve some domain. When we group by name, we see 16% of orphans and 17.6% of domains using them related to malicious activity. When we group by IP address, these are 26.1% and 43.8%.

6.6 Discussion

Our study identifies three reasons why orphans exist. These are typographical errors and misconfiguration in TLD zone files, malicious activity, and as placeholders for deleted domains. We also find that a significant number of the orphans are being used to send spam, and a smaller number of orphans are used for other malicious activity including hosting scam sites and serving as the DNS servers for phishing and malware sites. Although we only found a small amount of malicious activity related to orphans directly, by finding relationships among orphans and known malicious domains using them, we find evidence that a considerably greater number are involved.

Because orphans are sometimes used as placeholders for deleted domains, such domains will be incorrectly penalized by taking the presence of orphans as a sign of maliciousness. To account for false positives, the presence of strings such as `deleted` in the domain name appears to work as a good heuristic. However, since such placeholders are possible without the use of orphans, we instead recommend that registrars stop this practice. Additionally, a common reason for domains to be removed is take-down due to malicious use. If this is the case, other domains using the same DNS server may be malicious as well, justifying counting them as possibly malicious as well.

The overheads of determining if a web site is connected to an orphan DNS server are small for an end system or a monitoring system. The monitoring system must make a DNS request to find the DNS server for the domain in question. An end system requires this information anyway to find out the address of the web server to be visited. As soon as the DNS server is determined, an additional request could be made to determine if the server is in a domain that has an `NS` record. At an end system, this can be done in parallel with the request for the address of the web server, so the end user would not experience any additional delay.

7

Malicious Autonomous Systems

7.1 Introduction

Much of cyberfraud thrives on armies of compromised hosts, or *botnets*, which are scattered throughout the Internet. Contrary to what this may at first lead one to believe, malicious activity is not necessarily evenly distributed across the Internet: some networks may employ lax security, resulting in large populations of compromised machines, while others may tightly secure their network and not have any malicious activity. Further, some networks may exist solely to engage in malicious activity. Several recent Internet Service Provider (ISP) enforcements, such as the Atrivo and McColo Autonomous System (AS) de-peerings [31, 48] and the FTC closure of Pricewert networks [10], highlight that there are networks that exist simply to launch attacks.

In this chapter, we examine the AS distribution of malicious activity. This examination can be used to build metrics to identify ASes with disproportionately large amounts of malicious activity. Such a metric could be used as a feature in our framework, to judge if a web site is on a known malicious network.

To determine which ASes are malicious, we use ten of the most commonly-used blacklists for spam, phishing, malware and botnet activities for a period of a month, in addition to our `LocalSpam` dataset of URLs from spam collected at our department's email server. For host name-based blacklists, we first determine the IP addresses for each blocked host using Domain Name System (DNS)

queries. We then use Border Gateway Protocol (BGP) routing tables to group the IP addresses from each blacklist into their originating ASes. Upon grouping these addresses by AS, we compare ASes by the percent of infected machines. Using data from the RouteViews Project [76], we examine other characteristics of the malicious ASes, such as whether their connectivity to other ASes changes more often than those without malicious activity. The key findings in this chapter are:

- A large fraction of routable space is malicious for many ASes: Four ISPs each have over 80% of their routable IP addresses blacklisted. This raises red flags regarding the existence of such ISPs.
- Many ASes account for significantly large fractions of blacklists: Four ASes, three of which are US-based hosting providers, account for over 6% of at least one of the blacklists we tested.
- Many providers either harbor malicious activities or fail to consider it when peering: We find 22 provider ISPs with 100% of their customer ASes engaged in significant malicious activity.
- Malicious ASes differ from benign ones in other ways: They are more likely to become completely unreachable than those which have less malicious activity, and they are likely to have more peers.

Overall, these results confirm that some ASes contain disproportionately large amounts of malicious activity, indicating that a feature based on AS maliciousness could be useful.

7.2 Data Collection

To create a comprehensive evaluation of an AS, we use a diverse set of the data sources described in Chapter 3. These are summarized in Table 7.1.

For each data set, the data was collected from June 1, 2009 to June 30, 2009 unless otherwise indicated. We now discuss how each data source is specifically used in this chapter.

For phishing data, we use the `APWG` and `PhishTank` data feeds. On an hourly basis, we extract host names from the URLs currently in the feed, and perform DNS resolutions in each host name to

Label	Duration (days)	Unique IP Addresses	Unique ASes
APWG	30	9,560	1,803
Bot C&C	30	1,986	611
CleanMX	30	2,974	687
eSoft	30	8,000	1,196
Local Spam	30	5,495	1,024
Malware Patrol	30	871	368
PhishTank	28	7,143	1,580
Spamhaus SBL	29	6,422	2,005
Spamhaus XBL	29	29,585,604	13,580
SI-Feed	30	7,591	1,420
SI-DNS	30	4,448	911
SURBL	30	29,324	2,739

Table 7.1: Overview of Data Sets used for finding malicious ASes

get lists of IP addresses associated with these feeds. The PhishTank data set had a two-day outage on June 20 and June 21 causing us to only have 28 days of data for that data set.

We use the `SupportIntelligence`, `SURBL`, and `LocalSpam` lists of scam sites. Not every URL in `SupportIntelligence` has an IP address listed, so we use the ones that are listed as one set, `SI-Feed`, and perform our own resolutions as well and use them as another set, `SI-DNS`. We receive `SURBL` and `LocalSpam` data once per day, and `SupportIntelligence` once every six hours.

As a source for spam sender IP addresses, we use the `SBL` from Spamhaus. We also use the `XBL`, also from Spamhaus as a source of exploited machine IP addresses. The data collection for each of these starts a day later than the others, June 2, 2009. The `SBL` is updated every hour, the `XBL` every half-hour.

We use three feeds of web sites hosting malware, `eSoft`, `MalwarePatrol`, and `CleanMX`. We perform DNS resolutions on the host names extracted from these hourly to get IP addresses.

Finally, we use one data source, `ShadowServer`, as a source for botnet command and control servers. This list directly contains IP addresses, and we update it hourly.

7.2.1 Data Set Comparisons

When identifying malicious ASes for use as a feature, we would like to know if conclusions make about the maliciousness of each AS hold in general, or are specific to certain types of malicious

Number of Blacklists with Given IP Address	Number of IP Addresses
1	29,631,573
2	9,566
3	3,650
4	1,290
5	320
6	112
7	29
8	7
9	8

Table 7.2: Number of IP addresses appearing in multiple blacklists

activity. Due to differing goals, methodologies, and data sources, each data set we use can be expected to contain IP addresses not seen in the others. By examining the overlap of IP addresses from different data sets, we can see how often IP addresses are used for multiple different malicious purposes. We first examine if individual IP addresses are involved in multiple types of malicious activity, and then examine the same at the AS level.

In Table 7.2, we show the number of data sets containing each IP address. Intuitively, the `XBL` is three orders of magnitude larger than any other data set, so the vast majority of IP addresses appear only in a single that data set. It is further unsurprising that some IP addresses appear in two or three data sets since some of our data sets track the same information. We see that some IP addresses appeared in multiple data sets, with 8 IP addresses appearing in 9 of our data sets and another 7 appearing in 8 sets. This indicates that malicious machines are occasionally used for many forms of malicious activity, however a large majority appear not to be.

Now, we look at similarity between any two data sets. We calculate the Jaccard similarity coefficient between the sets of IP addresses in each. Let IP_{S_i} be the set of IP addresses in data set S_i. Then the Jaccard similarity of two data sets is given by $J(IP_{S_i}, IP_{S_j}) = \frac{|IP_{S_i} \cap IP_{S_j}|}{|IP_{S_i} \cup IP_{S_j}|}$. Results for all data sets except for `XBL` are shown in Table 7.3. We ignore the XBL because its size is orders of magnitude bigger than any other data set, hence the Jaccard coefficients involving it would be extremely small. As expected, we see the highest similarity between the two phishing data sets, and the two derived from `SupportIntelligence` data. Notably, the `ShadowServer` data set shares at most 4 IP addresses with any other data set, while most others, even measuring different types of

	Bot C&C	CleanMX	eSoft	Local Spam	Malware Patrol	Phishtank	Spamhaus SBL	SI-Feed	SI-DNS	SURBL
APWG	0	.06	.05	.02	.01	.24	.01	.04	.03	.10
Bot C&C		0	0	0	0	0	0	0	0	0
CleanMX			.07	.01	.06	.07	0	.01	.01	.02
eSoft				.01	.01	.05	0	.02	.01	.02
Local Spam					.01	.02	.01	.06	.09	.05
Malware Patrol						.01	0	.01	.01	.01
Phishtank							.01	.02	.02	.05
Spamhaus SBL								.01	.01	.01
SI-Feed									.49	.06
SI-DNS										.06

Table 7.3: Jaccard similarity between IP addresses in each data set

bad behavior, have greater similarity to each other. This exercise reinforces our belief that malicious machines are only occasionally used in multiple fraud campaigns, at least on smaller time scales, such as a month. Although one of the most similar, the two phishing data sets still only share 24% of their combined IP addresses with each other. The malware data sets have even less similarity in IP addresses. Obviously the individual malicious IP addresses are often not widely used enough or not in use long enough to capture the attention of multiple blacklists.

We now repeat a similar calculation for the overlap between the ASes represented in each data set. Let AS_{S_i} be the set of ASes represented in data set S_i. Then the Jaccard similarity of the ASes in the two data sets is given by $J(AS_{S_i}, AS_{S_j}) = \frac{|AS_{S_i} \cap AS_{S_j}|}{|AS_{S_i} \cup AS_{S_j}|}$. Results for this calculation are shown in Table 7.4. Between all pairs of data sets, there is much more similarity with regards to ASes than there was in terms of IP addresses. While the same IP address is not often used for multiple different malicious activities, multiple IP addresses in an AS appear to be used this way more often. The greater overlap in ASes used for different types of malicious activity further reinforces the benefits of searching for malicious ASes and the usefulness of this feature. Regardless of the type of malicious activity an AS was seen engaged in, such a record could be used to help determine if other later suspicious activities are truly malicious.

	Bot C&C	CleanMX	eSoft	Local Spam	Malware Patrol	Phishtank	Spamhaus SBL	SI-Feed	SI-DNS	SURBL
APWG	.17	.26	.34	.25	.14	.49	.26	.31	.24	.43
Bot C&C		.18	.17	.15	.16	.18	.14	.16	.15	.14
CleanMX			.35	.21	.25	.27	.17	.22	.22	.20
eSoft				.24	.20	.33	.23	.30	.25	.30
Local Spam					.17	.22	.20	.31	.33	.25
Malware Patrol						.16	.12	.15	.17	.12
Phishtank							.26	.29	.23	.38
Spamhaus SBL								.27	.20	.29
SI-Feed									.58	.33
SI-DNS										.26

Table 7.4: Jaccard similarity between ASes in each data set

7.3 Degree of Autonomous System Maliciousness

Starting from the IP addresses from our data sets, we determine the originating AS for each, and use this to group hosts at the AS granularity. In order to map IP addresses to an AS, we used a June 15, 2009 BGP routing table from the RouteViews Project [76]. We chose this date because it is in the middle of our data collection and is expected to give us the best estimate of the routing information from that duration.

We loaded each advertised BGP prefix and originating AS from the RouteViews data into a trie data structure commonly used by the routers in deciding the next interface to forward packets on and performed longest prefix matches on each IP address to determine the AS associated with the address. Using the AS information corresponding to each malicious IP address, we examined the extent of AS maliciousness from two perspectives: the percentage of the AS found to be blacklisted and the percentage of the blacklist each AS constitutes. We first describe both approaches and their results in detail. We then examine the temporal behavior of listed machines and the peering relationships of malicious networks.

7.3.1 Examination of ASes by Fraction of Advertised IP Space

Given the number of malicious IP addresses associated with an AS, the most straight-forward approach to evaluating the ASes for maliciousness would be to simply order the ASes by number of malicious IP addresses. However, such an analysis would penalize the larger AS: they simply have more addresses so they have more hosts that could be compromised and blacklisted. Accordingly we must consider the overall size of the AS as a factor when looking for ASes that are disproportionately bad.

There are no direct sources that help estimate the size of an AS. Even the *whois* database, which contains contact information about ASes in addition to detailed information about domain names and IP addresses, does not contain information about which AS owns which IP prefix. However, the prefixes advertised by an AS can be used to determine the maximum number of IP addresses associated with the AS. While ASes often have unused IP addresses in each of our prefixes, and it is difficult to determine just how many addresses are unused, this allows us to obtain a rough approximate for the AS size which may be considered an upper bound. We again extracted the prefix and originating AS information from the June 15, 2009 BGP routing table. We loaded this information into a trie data structure as before. For each prefix associated with an originating AS, this allowed us to determine the number of IP addresses associated with the prefix. In the process, we were careful to exclude any sub-prefixes associated with other ASes. Such a sub-prefix may exist, for example, if an ISP leases part of its address space to a customer with their own AS. After adding together the address space from each of the prefixes for each AS, we had the total number of IP addresses advertised by each AS.

With information about the number of unique machines found in the data feeds in each AS and the rough size of each AS, we can determine the rough percentage of each AS that appears in each data set. In Figure 7.1, we show the percentage of bad IP addresses in each AS present in our data sets, excluding the `XBL` data set. We separated out the `XBL` due to its much larger size which made the other results difficult to read. The AS indices on the x-axis do not match across data sets; it would be incorrect to think that the same AS exhibits a high percentage of maliciousness of all kinds. This figure shows several interesting results. First, a total of 31,263 ASes were advertised

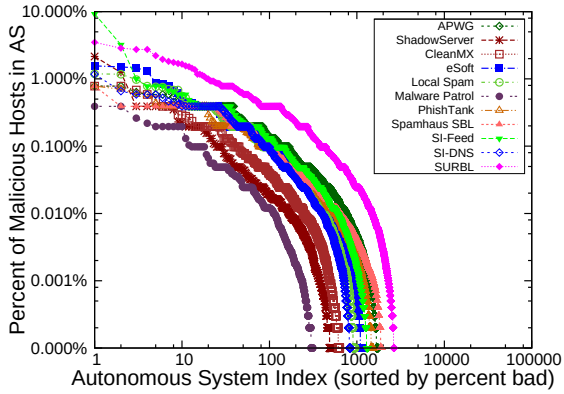


Figure 7.1: Percentage of bad IP addresses in each AS

in our BGP routing data and 46.8% of these had at least one malicious IP address in them. While a majority of them have little to no malicious activity, a small number of ASes have as much as 0.5-10% of their IP addresses engaged in malicious activities In fact, in the `SI-Feed` data set, one AS had 9.25% of its addresses in the data set. No other AS had 5% or more of its addresses in any of these data sets.

In Figure 7.2, we show the same results for the `XBL` data set and the combination of each data set. We note that the two lines are very similar and almost completely overlap because of the size of the `XBL` data set. We found 58 ASes with over 100,000 compromised machines in this data set. Additionally, 255 ASes had between 10,000 and 100,000 machines blacklisted. When looking at the percentage of each AS's advertised address space marked as malicious, we found that four ISPs, two from Ukraine, one from Iran, and one from Belarus, had at least 80% of their advertised IP space blacklisted. Another 49 in the `XBL` data set had 50-80% of their addresses listed. A further 556 ASes had at least 10% but fewer than 50% of their ASes listed. These ASes may have too lax a security policy or may be intentionally harboring cybercrime.

7.3.2 Examination of ASes by Proportion of Data Set

While examining the percentage of an AS that is blacklisted can highlight ASes with disproportionately high concentrations of blacklisted hosts, it requires large data sets. While this leads to good results based on the `XBL` data set, other data sets are not large enough to distinguish atypically

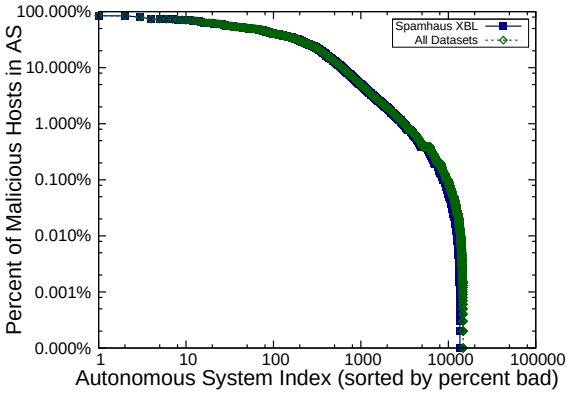


Figure 7.2: Percentage of bad IP addresses in each AS in the XBL blacklist and across all blacklists combined.

malicious networks. However, rather than consider the AS to be malicious based on the percentage of its blacklisted address space, we can instead examine the percentage of a data set that an AS accounts for. This can be used to highlight ASes with a large number of blacklisted hosts.

In Table 7.5, we show for each data set the number of ASes containing at least 0.25% of the IP addresses in the data set. This marks those ASes as bad with the highest absolute number of IP addresses flagged as malicious by each data set. However, in doing so, we wanted to avoid penalizing large ASes that advertise large address spaces and do not necessarily account for a disproportionate amount of maliciousness in that data set. Toward that goal, we perform the following calculations. Let AS_{S_i} be the set of ASes represented in data set S_i, and IP_{S_i} be the set of IP addresses in the data set. For each AS $a_j \in AS_{S_i}$, let IP_{a_j} be the set of IP addresses in the AS (without regards to whether the IP addresses themselves are in the data set). Then the IP addresses we count as malicious are those which satisfy the following two inequalities.

$$\frac{|IP_{a_j} \cap IP_{S_i}|}{|IP_{S_i}|} > .0025$$

$$\frac{|IP_{a_j}|}{\sum_{a_k \in AS_{S_i}} |IP_{a_k}|} * 10 < \frac{|IP_{a_j} \cap IP_{S_i}|}{|IP_{S_i}|}$$

The first of these inequalities simply captures those containing at least 0.25% of the IP addresses

	All	APWG	ShadowServer	CleanMX	eSoft	LocalSpam	MalwarePatrol	PhishTank	SBL	XBL	SI-Feed	SI-DNS	SURBL
$\geq 10\%$													
[9%, 10%)			1										
[8%, 9%)			1										
[7%, 8%)	1									1			
[6%, 7%)											1		
[5%, 6%)					1			1					
[4%, 5%)	1	1	1	2						1	1		1
[3%, 4%)	1					3	1			1		1	2
[2%, 3%)	3	2	2	2	3	2	1	1		3	1	2	
[1%, 2%)	7	5	5	3	7	11	6	3		7	5	10	8
[0.50%, 1%)	16	12	10	16	6	19	16	11		16	20	19	14
[0.25%, 0.50%)	19	20	26	27	25	20	18	18	18	18	27	33	38

Table 7.5: Number of ASes in each data set containing the given percentage of all IP addresses in the data set.

in the data set. The second ignores ASes where the proportion of the address space advertised by all ASes belonging to the data set advertised by the AS in question is greater than a factor of ten less than its proportion of the IP addresses. For example, if an AS contained exactly 0.25% of the IP addresses in the data set, we would list it if it accounted for under 0.025% of the address space of all ASes in the data set, but ignore it otherwise.

We can see that some ASes have a high concentrations of malicious activity. Focusing on the top few rows of Table 7.5, we note that several ASes account for more than 6% of blacklisted IP addresses in various data sets. For example, in the `ShadowServer` data set, we see that one AS contains 9.11% of the IP addresses in the data set, yet its advertised address space represents only 0.002% of the address space advertised by all ASes in the data set. The next AS in this list, with 8.66% of the listed IP addresses represents only 0.006% of the advertised addresses in the listed ASes. These two ASes are a large broadband ISP from Turkey and a hosting service provider from the US. Incidentally, the US-based hosting provider accounts for 7-8% of all blacklisted IP addresses also. Further, in `XBL` and `SI-Feed` data sets, we find two more US-based hosting providers that account for over 6-8% of these blacklists.

Overall, our results show that a small number of ASes have a disproportionate fraction of malicious hosts. These ASes may harbor malicious activity and perhaps should be investigated similarly to Atrivo or McColo [31,48]. We believe that legitimate ISPs with disproportionately high malicious activity need to provide tighter account controls, particularly in the case of hosting providers, or seek opportunities to provide anti-virus or firewalling services to prevent malicious activity.

7.3.3 How Long Do Hosts Stay Infected?

One can argue that ASes that clean their infected hosts are more responsible than those that do not. Unfortunately, it is difficult to remotely confirm whether a machine has been cleaned or that it is simply (temporarily) not being used for malicious activity. The best we can do is to determine whether the host exhibits the same malicious behavior and is captured by our data set for long time periods. Unfortunately, this analysis is limited by the manner in which blacklists are maintained. In order to exclude blacklists that may not be purging clean machines quickly, we exclude the two Spamhaus data feeds because the data sets are large and entries are only removed manually. Using the remaining feeds, we examine how long IP addresses are listed in the data set.

In Figure 7.3, we plot the percentage of IP addresses in each data set listed for a given number of days. Overall, we find that either IP addresses are listed in a blacklist for under five days or they are listed for over 25 days and sometimes for the duration of our data set. This indicates that some ISPs may either be lax about cleaning up their machines or may intentionally choose not to do so.

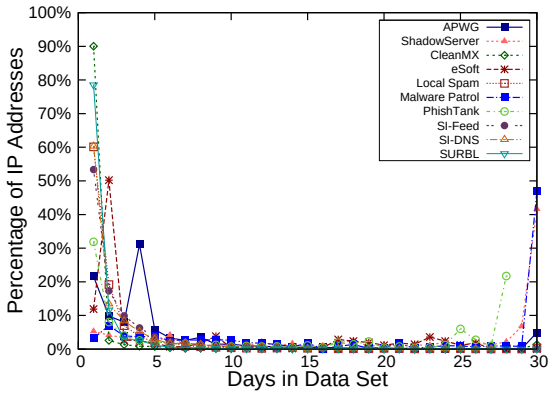


Figure 7.3: Percentage of IP addresses that remain in the indicated data set for the given duration.

In Figure 7.4, we show the number of days on which an AS had at least one host in the data set. While the graph is similar to Figure 7.3, there are two important differences. First, we find that smaller percentage of ASes have their IP addresses listed on 5 days or less. Also, a larger fraction of ASes have their IP addresses listed for 25 days or more.

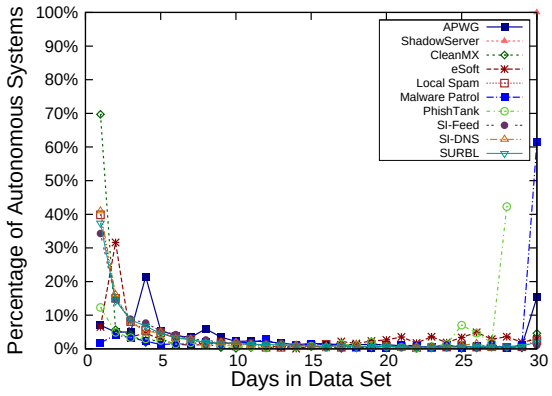


Figure 7.4: Percentage of ASes that remain in the indicated data set for the given duration.

7.3.4 ASes with Unruly Children

Our data establishes that malicious activity is often disproportionately contained in certain ASes. We now examine whether ASes with disproportionate malicious activity are closely related. We begin by labeling as malicious any AS with at least 1% of its IP addresses appearing in any blacklist, as described in Section 7.3.1. We then examine each of the BGP updates for June 2009 to find provider-customer (or parent-child) relationships. Given two adjacent ASes, we infer which one is the parent by examining the degrees of the two ASes, similar to the algorithm described by Gao [21]. We consider the AS with largest degree to be the provider.

For each provider AS, we consider the extent to which its customer ASes have been found to be malicious. In the second column of Table 7.6, we show the number of provider ASes with at least three children that have the indicated percentage of its children as malicious. We see 22 ASes with 100% of their customers classified as malicious. A total of 194 providers have at least 50% malicious customer ASes.

We repeated this analysis using the definition of maliciousness from Section 7.3.2: the AS must

Percent of Malicious Customer ASes	Number of Provider ASes	
	Fraction of Advertised IP Space	Proportion of Data Set
100%	22	
[90%, 100%)	2	
[80%, 90%)	8	
[70%, 80%)	17	
[60%, 70%)	72	3
[50%, 60%)	73	2
[40%, 50%)	78	5
[30%, 40%)	202	24
[20%, 30%)	239	45
[10%, 20%)	204	78

Table 7.6: Percentage of malicious customer ASes for providers with more than three customers.

have at least 0.25% of the malicious IP addresses in a data set. We show these results in the third column of Table 7.6. Five providers have at least 50% of their customer ASes labeled as malicious.

This analysis shows that there are dense clusters of malicious activity in the Internet. This may be an indication that there are upstream providers that are willing to peer with any customer, regardless of whether it harbors malicious activity. We hope that studies similar to ours would put pressure on provider ASes to extensive screen their customers and require their customers to limit malicious activity as part of their peering agreements.

7.4 Identifying Malicious Autonomous System

Having examined the degree of AS malicious behavior, we now search for other characteristics that differ between malicious and benign ASes. Specifically, we compare ASes where we have not observed any malicious IP addresses (good ASes), ASes where we have seen at least one malicious IP address, ASes which have at least 1% of their IP addresses in one of our malicious data sets, and ASes representing at least 0.25% of a blacklist as described in Sections 7.3.1 and 7.3.2. For these categories, we compare BGP behavior, AS size, and AS connectivity. ASes can be disproportionately malicious for several reasons, such as malicious intent by the operator of the AS, or just lax administration practices. Therefore, we do not expect all malicious ASes to have the same properties as each other

or for there to be no overlap with good ASes. However, we do hope to see trends in the characteristics of the malicious ASes.

7.4.1 BGP Behavior

In order to examine BGP behavior, we begin with the earliest BGP routing table available from the RouteViews project for June 1, 2009. We then replay in order all of the BGP updates for the month of June, examining how routes change in the updates.

We begin by examining routing changes that result in any AS which originates a prefix becoming completely unreachable. We consider an AS to become unreachable when all of the routes to all of the prefixes originated by that AS have been withdrawn according to all of the routers that peer with RouteViews. In total, 5,069 ASes become unreachable at some point in the month. This is 15.7% of the 32,193 total ASes we ever see originating a route.

In our data sets of malicious activity, we observed IP addresses from 14,807 ASes. Of these, 2,319 become unreachable at some point. This is the same percentage, 15.7%, that became unreachable when examining all ASes. It appears that the chances of becoming completely disconnected or unreachable is not affected by the degree of maliciousness contained. However, looking at just those ASes where 1% of their IP addresses have been marked as bad, we see that 24.4% become unreachable. ASes with the most malicious activity appear to disconnected more often than others. Among the ASes which make up at least 0.25% of the malicious IP addresses in their data sets, only 8 (3.0%) ever become unreachable.

Many of the ASes which become unreachable do not stay that way for long. We now look at if how long they are unreachable is dependent on the degree of maliciousness of the AS. Figure 7.5 shows the duration of time ASes in each category become unreachable, except for those making up at least 0.25% of malicious IP addresses in a data set, which we exclude from this figure due to the low number of data points. Some become unreachable multiple times for short durations; however, the time plotted in this figure represents the aggregate for each AS. Timestamps on the BGP updates are at a resolution of one second, so when an AS becomes unreachable for shorter than one second, we count it as becoming unreachable but do not add time for this period.

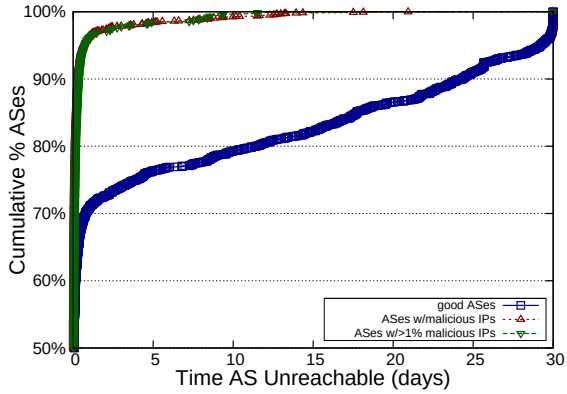


Figure 7.5: Unreachability duration for good and bad ASes which become unreachable during our data period.

We see a significant difference here between our categories. 96% of malicious ASes are disconnected for shorter than a single day, with a similar number for ASes with 1% bad IP addresses. However, this is only 71% for the ASes not identified as malicious which become disconnected. On the high end, while 45.7% of ASes which become unreachable have malicious behaviors, just 1% of those unreachable for more than 2 weeks have malicious behaviors. When malicious ASes become unreachable, they do not tend to stay that way for long. If these disconnections are intentional de-peerings, the approach is not effective at isolating the AS for long.

The results for the length of time an AS becomes unreachable were opposite of what we initially expected. To examine routing behavior in further detail, we now consider all connectivity changes to ASes which originate a route (gaining or losing a peer), not just those which change its overall reachability. Of all ASes originating a prefix, 17,286 (53.7%) have some change during our data period. For malicious ASes, this is 8,695 (58.7%), and for those with at least 1% malicious IP addresses, this is 2,036 (66.1%). For those making up at least .25% of one of our data sets, this is 166 (60.9%). Malicious behavior in an AS is clearly associated with routing instability; however, this may not be a causal relationship. Both could be the consequence of poor management.

The presence of connectivity changes may be due to problems with the other peer involved in the connection. This is less likely to be the explanation for such changes, if an AS had such changes in its relationships with more than a single peer. Figure 7.6 shows the number of peers involved in connectivity changes with each AS that had such changes. Among good ASes, only 36% with

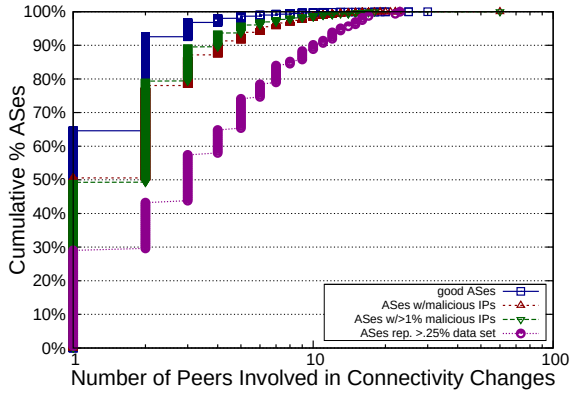


Figure 7.6: Number of peers involved in connectivity changes for each origin AS with such changes in our data period.

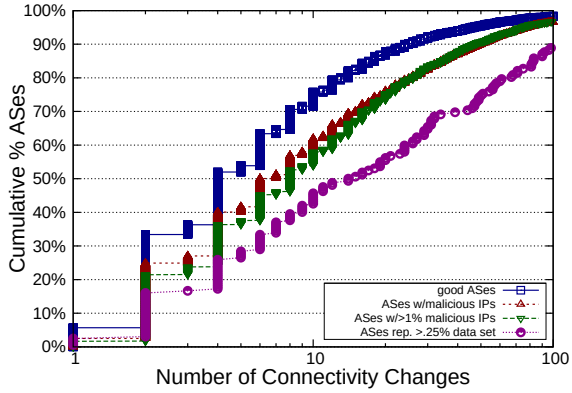


Figure 7.7: Number of connectivity changes for each origin AS with such changes in our data period.

changes had connectivity changes with multiple peers. However, among bad ASes, this is much higher: 50% had a change in relation to more than one peer. This was similar for those with more than 1% bad IP addresses, but was worse for those ASes making up at least 0.25% of their data set. For these, 70% changed in relation to more than a single peer.

Similarly, Figure 7.7 shows the total number of connectivity changes. Among good ASes, 75% of those with changes had 10 or fewer total changes, while this was only 62% for bad ASes and 45% for bad ASes representing 0.25% of their data set. Overall, among those with changes, ASes harboring malicious behavior have a greater number of connectivity changes than good ASes, and these changes involve more of their peers.

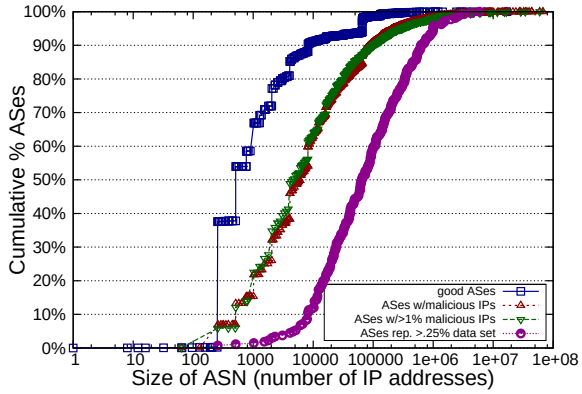


Figure 7.8: Sizes of ASes containing or not containing malicious IP addresses in our blacklists.

We note that some ASes neighboring those who harbor maliciousness seem to be taking some action against such behavior, since malicious ASes are more likely to be involved in a greater number of routing changes than others. However, this does not seem to often result in overall losses of connectivity for the ASes with malicious behavior. Alternatively, problems with managing the AS may be to blame, with instability in connectivity and presence of malicious activity both being symptoms of a larger problem.

7.4.2 AS Sizes

We now investigate whether bad ASes have differing sizes than good ones, to see if either larger ASes or smaller ASes have a greater tendency towards malicious behaviors. For each AS, we use the BGP routing table from June 15 to determine the size of the AS based on the size of the prefixes they advertise. Results are plotted for our four categories in Figure 7.8.

We see significant difference between the sizes of good ASes and those containing malicious IP addresses. While the median size for a good AS is 512 IP addresses, the median for ASes with any malicious IP addresses at all and those with more than 1% of their IP addresses malicious is an order of magnitude larger, and the median for those that represented more than 0.25% of a data set is yet another order of magnitude larger. Similarly, while 67% of ASes without malicious IP addresses have 1024 or fewer IP addresses, this is only 22% for those containing malicious IP addresses, and 1.5% for those that made up at least 0.25% of a data set.

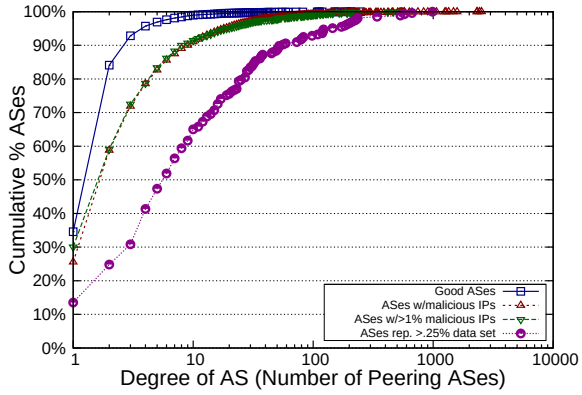


Figure 7.9: Degrees of ASes containing or not containing malicious IP addresses in our blacklists.

This result is to some extent expected. The more addresses in an AS, the more likely at least one will be compromised. However, the plot for those with more than 1% of their addresses marked as malicious closely follows the plot for those with any malicious addresses at all. This is unexpected because larger ASes would need more total IP addresses to be malicious to end up in this category. Overall, it appears that larger ASes are more likely to contain malicious addresses. One explanation is that small ASes may be centrally administrated, yielding stronger security guarantees while larger networks may have heterogeneous administration leading to more opportunities for lax security.

7.4.3 Degree of AS Peering

We now look at the degree of each AS, which is the number of other ASes with which it directly connects. We would expect for ASes containing malicious IP addresses, especially in large proportions, to have a lower degree because others would be less willing to peer with them. However, this turns out not to be the case, as we show in Figure 7.9.

Figure 7.9 shows that the ASes with malicious IP addresses are more likely to have a higher degree. Both have a median degree between 1 and 2 indicating that a large portion of both are stub ASes. 99% of good ASes have a degree of 10 or below, while this is 91% for ASes with at least one malicious host, and only 65% for ASes with at least 0.25% of the malicious IP addresses in a data set. ASes containing malicious behavior may peer with more neighbors in case one disconnects them, or they may have a higher degree because they are large ASes which lack centralized administration

and lead to increased malicious behavior. In any case, the higher degrees of ASes with malicious IP addresses may explain why they have more peering changes than good ones but tend not to lose connectivity for long periods of time.

7.5 Discussion

In this chapter, we examined whether some networks are safe harbors for malicious activity. We found that several ASes have high concentrations of malicious IP addresses while others represent disproportionately higher malicious activity than their equivalently sized peers. This shows that while botnets are commonly being used to launch attacks, malicious hosts may still be clumped by network providers. In spite of these results, traffic cannot simply be declared malicious based solely on its originating AS even for ASes with the high degree of maliciousness, as this would have extensive collateral damage, penalizing legitimate traffic as well. However, if they are exhibiting the other features of malicious activity described in this dissertation, we can be more certain the traffic is malicious.

Determining if an AS is malicious requires extensive blacklists and knowledge of the size of an AS which is most easily derived from BGP tables. A monitoring system could compile this information itself and maintain a list of malicious ASes. However, this information is not commonly stored on end hosts, and systems to easily query it are not in place. Therefore, if implemented on end hosts, it would not be practical for them to compile this information individually. Instead, AS reputations would have to be calculated on a periodic basis by some central entity with access to a large amount of blacklist data, and made available for end systems to issue queries to. Once the list of malicious ASes is generated, it is easy for a monitoring system or end hosts to determine what ASes a given IP address is in. This simply requires a single longest prefix match operation.

8

Web Redirects

8.1 Introduction

While browsing the web, users often find themselves redirected to a Uniform Resource Locator (URL) other than the one they clicked on. Such redirects have several uses. For example, site operators may use redirects is to track their users' browsing patterns. Normally, a site will not be able to tell which of the external links a user followed from their web page. However, if the link contains a redirect which causes the user to contact the site's web server first, the server can record user activity and then provide the URL for the new destination. Other uses include directing users to the correct new URLs in the case of a web site redesign, bringing users from a misspelled version of a domain name to the correct one, and shortening long URLs so they are easier to share with others.

From a security viewpoint, one very interesting type of redirects are the ones that are *open*. Links containing redirects often use parameters to control the destination of the redirect. If the web server does not check the parameters appropriately before taking action, one can manipulate the destination. This seemingly inconsequential aspect of open redirects has been abused in phishing attacks [20, 70]. For example, assume the following URLs is a redirect, leading the user to `www.cs.indiana.edu`: `http://example.com/redirect.php?dest=http://www.cs.indiana.edu/`. This redirect would be open if a phisher could replace `www.cs.indiana.edu` with any host name or IP they desired and the `redirect.php` script on `example.com` would direct the browser to the new URL.

Phishers can abuse open redirects on trusted sites to mislead users about the site they are visiting. A casual Internet user who receives this link, perhaps in an email, may think that they were visiting `example.com` when in reality they was only being redirected by `example.com` to the phisher's domain, especially if the phisher's site had a look and feel designed to imitate that of `example.com`.

In this chapter, we take two approaches to redirects. First, we examine the prevalence of open redirects on the web. Next, we compare the use of redirects (open or not) by known phishing sites to their use by legitimate web sites. We refer to these redirects in use by malicious web sites as *malicious redirects* in this chapter. Although we do not explicitly study redirects used by scam or malware sites, they would be useful to miscreants for drawing traffic to these sites just as they are in phishing, and therefore useful as a feature for finding these types of malicious web sites. We arrive at the following key results:

- Open Redirect Prevalence: We find that a significant proportion of redirects on the web are open. 79% of redirects containing the destination in the URL were open.
- Malicious Redirects: In our preliminary study of malicious redirects, we find 75% of URLs from a feed of phishing sites contain redirects. In contrast, only 12% of URLs from a directory of legitimate sites contained redirects.

8.2 Background

A variety of mechanisms can be used to implement redirects. One technique, highly recommended by the World Wide Web Consortium (W3C) [115], is the *HTTP redirect*. It uses the HTTP protocol to redirect users. When a user clicks on a link containing an HTTP redirect, the web server responds with the URL for the destination of the redirect along with a status code indicating to the user that she is being redirected.

Another approach exploits the *HTML refresh* to redirect users. The refresh capability exists primarily to allow sites whose content changes frequently to specify in the HTML code of their

pages how often the pages should be automatically reloaded. A redirect exploiting this mechanism basically leverages the fact that it allows the destination of the reload to be specified. Since this destination could be different from the page the user is visiting, it can be used to redirect users.

HTTP redirects may be either static or dynamic. Links containing static redirects always lead to the same destination. Links containing dynamic redirects often embed the destination of the redirect in a query string contained in the URL itself. When a user clicks on the link, the browser sends the query string with the request. The web server reads the parameters contained in the query string to decide the destination of the redirect. Dynamic redirects offer greater flexibility and convenience for the web site operator, but can be abused if not properly secured. Due to the potential for abuse of this dynamic capability, we focus on HTTP redirects.

A typical link containing a dynamic redirect has the following structure. In the URL, `http://example.com/redirect.php?dest=http://1.2.3.4/`, `example.com` is the name of the server the client contacts and `redirect.php` is the script the server runs. The script takes the query string, `dest=http://1.2.3.4/`, as parameters. The query string starts with a "`?`" character and is frequently composed of a series of name and value pairs delimited by the "`&`" character. In this example, the value, `http://1.2.3.4/`, is associated with the name `dest`.

8.3 Open Redirects

We first focus on open redirects. We begin by developing heuristics to identify open redirects. Then we investigate the prevalence of such redirects.

8.3.1 Heuristics to Identify Open Redirects

Given a particular link, our heuristics proceed in three stages. They determine 1) if it contains a redirect, 2) if the redirect is open, and 3) if it is not, whether it can easily be modified to be open.

Heuristics to Find Redirects

Dynamic redirects are the only redirects which may be open. Our heuristic to find dynamic redirects is the following: we search for the presence of a query string in the URLs. In the URLs that contain a query string, we search for protocol prefixes `http://` or `https://`, either in plain text or encoded, which signal the destination of the redirect. We consider the link to contain a potential redirect if a query string is found containing a URL pattern.

A parameter containing a URL may be present for other reasons. It does not necessarily mean the link is a redirect. Therefore, it is necessary to validate if what our heuristic considers a potential redirect is indeed a redirect. First, we access the URL, automatically following HTTP-based redirects, allowing us to determine the URL of the page at the end of the redirection chain. If the URL of the final page is different from the original URL requested, we classify the page as using a redirect. The rest are not redirects.

We also note that a redirect does not necessarily mean that the destination of the redirect matches the destination contained in the query string of the URL. One such case where this may happen is when multiple cascaded redirects occur. Since we are interested in determining whether it is possible to manipulate the destination of the redirect for attacks, we focus only on the redirects where the final destination matches the destination contained in the query string of the original URL. Subsequently, we refer to such redirects as *simple redirects*. To find these redirects, we first extract the destination of the redirect from the URL's query string. The destination begins with an `http://` or `https://` and either ends with a "`&`" character, which marks the beginning of the next key-value pair, or when the URL itself ends. Upon traversing the URL, if the final destination URL matches that contained in the redirect, we test it for openness. For example, if the redirect `http://example.com/redirect.php?dest=http://1.2.3.4/` resulted in the final destination URL of `\url{http://1.2.3.4/}`, we scrutinize it further; otherwise, we exclude it from further analysis.

Heuristic to Find Open Redirects

A redirect is *open* if the destination contained in its query string can be altered and the web server processing the redirect sends the client to the new location without validation. To test if a simple

redirect is open, we replace the destination URL contained in the query string of the redirect with a web site that we control. On that site, we include a randomly generated 200 character string that is unlikely to appear on other web pages. We then follow the link to determine whether it causes the browser to return a page containing the string. If so, we consider the page to be an open redirect.

Heuristics to Pry Redirects Open

If a redirect employs weak protections, it may be possible to pry it open. For example, some redirects may employ a checksum for the destination contained in the query string, preventing the redirect from being used if the checksum is incorrect. By altering the checksum along with the destination, one may be able to pry open such redirects. Though exploiting such redirects would require some thought on the part of the attacker, they can be easily exploited by others once an algorithm to open them is developed. We now explain the heuristics we use to test if a redirect which was not open according to the previous heuristic can be pried open.

Redirects which have query strings with only one parameter, the URL of the destination, clearly do not have other query string parameters securing the redirect. If they are not open, they must be using some internal mechanism, such as a white list, to secure the redirect. Such redirects cannot be pried open externally. For such cases, we focus on detecting whether such redirects use white lists containing popular web sites.

Redirects whose query strings have at least one parameter other than the destination required some thought. If altering the destination and leaving the remaining parameters unchanged failed to open the redirect, the redirect is secured internally or there is at least one parameter the server is testing before redirecting. Even though 90% of the redirects had 4 or few parameters including the destination parameter, varying each to infer which of the parameters could be altered would have been cumbersome.

Upon manual inspection, we found that some of the parameters are unlikely to be specific to the destination of the redirect. For example, two of the common parameters were related to language of the page and country of origin of the request. However, many sites will have the same language and country or origin.

Instead of trying to infer the intent of all parameters to check which ones were specific to the destination, we tried two very simple strategies: we either dropped all the parameters other than the URL, or altered each of them simultaneously in trivial ways. Specifically, if a parameter was a number, we simply incremented it, and if it was a string, we dropped a character from the string. Doing so essentially only checked if the server was checking anything at all for the altered destination. While one may expect that anything outside of the permitted destinations would be denied if the default case was handled properly, we found quite the contrary: many servers were only allowing the permitted destination with a given set of parameters, but allowed arbitrary destinations when these parameters were altered. We describe these and other results in Section 8.3.3.

8.3.2 Data Collection

To find the prevalence of exploitable redirects in the web, we performed extensive web crawls using three different data sets, described in more detail in Chapter 3. For each data set, we obtained a URL for a top page, downloaded that page and any page linked from that page that was within the same DNS domain as the original page. We examined all the links contained in the top-level pages as well as on the pages we followed using the heuristics described in Section 8.3.1.

In our first data set we extracted the $1,000$ most popular sites in each of 16 top level categories from `Alexa`, and the top 500 overall. Some sites were present in multiple categories; upon removing duplicates, we found $15,341$ unique web sites. We used each of the sites obtained from `Alexa` as starting pages for web crawling. This crawl resulted in $864,628$ web pages.

The second data set uses links from `DMOZ`. To obtain a similar number of sites as in the first data set, we randomly selected $16,500$ unique links in the `DMOZ` directory on October 23, 2007. We obtained $216,812$ web pages from this crawl.

For our third data set we used our `LocalDNS` data in order to focus on actual user behavior. In a one week snapshot, this data contained $164,145$ unique host names. From this crawl, we obtained $1,368,198$ web pages.

Data Set	Source Pages	Total Links	No Query String	No URL Pattern	Potential Redirects
Alexa	864,628	53,833,400	71.00%	27.92%	1.07%
DMOZ	216,812	5,745,145	70.09%	28.99%	0.92%
LocalDNS	1,368,198	81,186,127	73.07%	25.24%	1.70%

Table 8.1: Classification of links extracted from each data set

	Potential Redirects	Actual	Not Redirects	Broken
Alexa	283,001	68.75%	23.01%	8.24%
DMOZ	20,364	58.61%	19.28%	22.10%
LocalDNS	562,118	68.48%	18.17%	13.35%

Table 8.2: Classification of potential redirects from each data set

8.3.3 Prevalence of Open Redirects

Recall from Section 8.3.1 that potential redirects specify a destination in the URL's query string (identified by the presence of `http://` or `https://`). When eliminating URLs that fail these tests, between 0.92% and 1.7% of the links remained in our data sets, as shown in Table 8.1. We note that even links that are not regarded as potential redirects could be involved in redirects; these pages may use a mechanism to obfuscate their functionality. Short of traversing each of them individually, there is no way to find such redirects. Since visiting over 140 million links from our three data sets would have been very time consuming, we simply excluded these cases which did not have a query string or a destination specified. Accordingly, the results we present serve as a lower bound on the actual number of redirects on the web.

After removing duplicates there were $815,779$ unique potential redirect URLs. These spanned $4,978$ unique domains, and 82 unique Top Level Domains (TLDs). Validation of these redirects through an actual traversal, as described in Section 8.3.1, confirmed that $557,646$ (68%) were actual redirects. A further $100,191$ (12%) of the links were broken and could not be retrieved. The rest did not appear to use redirection even though they contained a query string with a URL pattern. Table 8.2 shows the breakdown of these categories by data set. We see that a significantly lower portion of potential redirects are actual redirects in the `DMOZ` data set, which is composed of random sites, than in the other data sets.

Recall from Section 8.3.1 that *simple redirects* are easily manipulated by attackers. These are

	Simple Redirects	Open	Closed	Broken
Alexa	65,012	83.00%	12.97%	4.03%
DMOZ	3,117	81.30%	13.86%	4.84%
LocalDNS	98,138	77.65%	19.65%	2.70%

Table 8.3: Classification of simple redirects from each data set

redirects where the destination of the redirect is the same as the URL included in the parameters of the query string. Of the actual redirects, 177,284 (32%) passed this test. Based on a manual inspection, those that did not fell into several categories: redirection to site authentication pages, transitions of the protocol from HTTP to HTTPS, search pages, and blog posting pages. While these links do have query strings that contain URL patterns and use redirection, these factors are independent, suggesting these redirects were statically configured even though they looked like dynamic redirects. Since these cannot be manipulated, we exclude these links from testing for open redirects.

Next, we tested if the 177,284 simple redirects were open, as described in Section 8.3.1. Replacing the destination of the redirect with our custom page was not possible for 16,142 of the URLs because they used non-standard character encoding; we excluded these redirects from subsequent analysis. Of the remaining 161,142 entries, 128,058 (79%) of the redirects were completely open: traversing them caused the server to redirect to our custom page instead of the original one contained in the destination of the redirect. Another 5,108 (3%) returned an error. From this, it is clear that sites that use parameters to determine the destination of the redirect fail to secure their redirects a vast majority of the time. These results are shown by individual data sets in Table 8.3. Results across the data sets were similar, with popular `Alexa` sites containing the highest percentage of open redirects. The `LocalDNS` data set had the lowest percentage of open redirects.

We then tested if the remaining simple redirects, 27,976 (17%), could be pried open. As described in Section 8.3.1, we used three approaches to subvert these protections: alter all parameters simultaneously except the URL (which reflects the new destination), drop all the parameters except the URL, and replace the redirect URL with a popular site possibly on the server's white list (we used `google.com`). To our surprise, 2,346 (8.4%) could be pried open by at least one of these approaches. Dropping the non-URL parameters was the only effective approach in 965 cases. Altering

the non-URL parameters was the only effective approach in 682. In 699 cases, either dropping or altering the non-URL parameters successfully resulted in prying open the redirect. There were no cases where simply changing the URL to a popular site opened the redirect. In total, simple redirects were either open or pried open in 81% of the cases we examined, yielding 130,404 unique open redirect URLs.

8.4 Malicious Redirects

While open redirects may be attractive to miscreants, we do not know how often they actually take advantage of them. Here, we present a preliminary look at the actual use of redirects by miscreants.

8.4.1 Data Collection

We use two sources of data for this investigation. For malicious URLs we use a snapshot of the `APWG` feed on June 15, 2010. This feed contains links to phishing sites. Trying to convince a user to click on a link leading to a phishing site is one place miscreants are likely to use misleading redirects. At the time we use it, this snapshot is under two hours old. For benign URLs we once again use the `DMOZ` data set, this time using a snapshot from May 26, 2010. Although this snapshot is older than the `APWG` snapshot we used, this should not cause problems as benign sites are relatively stable. Unlike when we looked for open redirects, here we directly use the URLs in the `DMOZ` data set, instead of crawling based on them. This is so our methodology for good sites is as similar as possible to our methodology for malicious. We use the entire snapshot of the `APWG` feed. However, since the `DMOZ` feed is so much larger, we sample it randomly at a rate of 0.1%.

Similar to our search for open redirects, here we continue to focus on HTTP redirects. For each URLs in the sample, we perform an HTTP HEAD request. This request is similar to the GET request used to retrieve a web page, however it does not return the page contents. The header, including the return code that tells us if we are being redirected, remains the same. We set our user agent string to match that of Firefox 3.0, in case some sites discriminated based on browser version.

	Total URLs	Without Errors	Percent Redirects
APWG	2401	1567	75.1%
DMOZ	4042	3922	12.1%

Table 8.4: Redirects found in benign and malicious URLs

8.4.2 Results

In total, we tested 2401 URLs from the APWG list and 4042 from DMOZ. Not all of these are valid. Some return server errors or error messages indicating the page no longer exists. This is especially likely to happen for malicious pages, as they may be taken down soon after discovery. Upon removing these, 1567 from APWG and 3922 from DMOZ remain. The results we present in Table 8.4 are in relation to these numbers.

Both malicious and benign URLs have a significant number of redirects. However, there is a large difference with over three quarters of APWG URLs using redirects, compared to a much smaller percent of DMOZ. When we see redirects, there are further differences. Among the DMOZ redirects, 52.2% used HTTP status code 301, meant to be used when a page has been removed permanently. In contrast, very few of the malicious pages used this code. Instead, 97% of them returned HTTP status 302, meant to be used when a page has been moved temporarily. A more detailed study of the characteristics of benign and malicious redirects is needed.

8.5 Discussion

Our analysis found that a large proportion of redirects on the web are open and can be manipulated and exploited. Although we do not know at this point if miscreants are taking advantage of open redirects, we know they are using redirects heavily. Although more work is needed, it appears that the mere presence of redirects can be a useful feature to identify malicious activity, especially in conjunction with other features discussed in previous chapters.

Ideally, the open redirect problem would be solved by careful design and configuration of software on the server side. Administrators could configure whitelists of approved destinations to ensure

their redirects are not misused. However, this approach requires the site to store a database of valid third-party redirect destinations, which may be extensive in some cases, and requires administrative overhead to keep the list current. Even if most server administrators closed their redirects, all it takes is a few open ones for miscreants to take advantage of. Therefore, client-side solutions are necessary.

At the end system, client web browsers can apply the heuristics we used to identify HTTP redirects in general or open redirects. To identify HTTP redirects there browser simply needs to check the status code returned from a HTTP request. This adds no overhead, since it must do this anyway to follow redirects. To identify open redirects, the browser would examine the link for any URL patterns in the query string. If a destination is specified, the browser would replace the URL pattern with a test verification web page. If upon following that link, the client is redirected to the verification page, it has confirmed that the redirect is open. This can be done in parallel to loading the users intended destination. A monitoring system could perform these tests as well. It would need to request each URL, but this feature should not have the overhead of a web crawler since page content is unnecessary, so HTTP HEAD requests could be used.

9

Related Work

Much previous work has been put into the identification of malicious activities on the Internet. We first discuss work related to each of the five infrastructure features we examine. Then, we discuss other work that examines characteristics of malicious infrastructure, some of which could be adapted to work in our framework. Finally, we briefly discuss other approaches to the scam, phishing, and malware problems.

9.1 Our Features

9.1.1 Fast Flux

The Honeynet Project and Research Alliance was the first to recognize the prevalence of fast flux in hosting malicious sites. Although their white paper [104] does not provide a model which can be used to identify fast flux, it provides several pieces of valuable information including two real world examples of Domain Name System (DNS) resolutions for fast flux host names, and a case study of the activities of an infected system. In a more recent study of similar nature, Konte *et al.* [47] examined the role of fast flux in hosting 21 online scams observed at a single spam trap.

Many fast flux campaigns are hosted on botnets. Nazario *et al.* [71] investigated the behaviors of botnets behind fast flux. They find 80% of fast flux domains are registered at least a month before

actual use. They also find a long lifetime for fast flux domains, a median of 18.5 days after the domain has become actively used.

The general topic of fast flux detection has also received attention from several researchers. Holz *et al.* [30] examine flux in scam sites. They produce a model to identify flux based on the number of DNS address records, Autonomous Systems (ASes), and DNS name server records for a host. They find about 30% of domains they collected from spam mails are using fast-flux. They always use information from two DNS resolutions, so their method of detection is not light-weight enough for use in our framework. In another similar study, Passerini *et al.* [80] also examine fast flux in scam sites. They crate a model to classify sites as fluxing or not based on nine parameters. Some of these are DNS measurements, some are similar to those we use, and several others are based on `whois` [13] records, including the age of the domain and its registrar. They do not investigate which of these features is actually effective. While we could use `whois`-based features for either flux detection or as features on their own, We choose to avoid these due to concerns surrounding the accuracy of `whois` information [34].

Caglayan *et al.* [8] developed a Fast Flux Monitor to detect fast flux in real time. Their system uses the Time to Live (TTL) on DNS records, along with scores quantifying the change in `A` records in a 10 minute interval, and the dispersion of the IP addresses. Although they have implemented a prototype of their system, they do not give a detailed evaluation of its effectiveness or what parameters are most useful in detection.

A different approach to combating flux is taken by Bambenek [7]. Instead of detecting it, he proposes making modifications to the DNS system to make it more difficult. Specifically, he proposes that the domain registrars limit changes to authoritative DNS servers to once every 72 hours. He also recommends that DNS servers not allow TTLs shorter than 24 hours for `NS` records. Further, he recommends that DNS clients not accept records with TTLs shorterthan 12 hours. Unfortunately, this proposal would be difficult to implement. It would require changes to DNS client software on every computer. The changes to servers would likely be ineffective due to miscreants running their own DNS servers without the changes. Additionally, these changes may have harmful effects on legitimate records, such as those resulting from Content Distribution Networks (CDNs), which also rely on low TTLs.

9.1.2 DNS Wildcards

Wildcard records have been a part of DNS from the original specification [63]. The description of wildcard behavior in this specification is ambiguous and unintuitive, so RFC 4592 [50] was created to clarify the intended behaviors of wildcard records.

In addition to issues arising from the specification being non-intuitive, it has been argued that they violate common assumptions on how DNS should operate. An Internet Architecture Board (IAB) commentary [33] describes the way wildcards violate these assumptions and the issues that can arise from it. It recommends only using wildcards with `MX` DNS records, since they are the only ones that only affect a single protocol. It also recommends not ever using wildcards for domains that have subdomains. Nonetheless, wildcards are in widespread use.

Other work points to evidence of wildcards being used for somewhat malicious behaviors. Two advisories by Netcraft [62,69] detail specific ways wildcards are used by phishers. They find wildcards in use for two purposes, first, to make the host name appear legitimate, and second, to randomize the host name to avoid blocking mechanisms. Zdrnja *et al.* [118] examine DNS responses collected at a university Internet gateway. They find many typo-squatted domains using wildcards. While they claim the risk from such domains is high, they do not specifically find them involved in malicious activities.

9.1.3 Orphan DNS Servers

To our knowledge, there has been only a single other work that has looked into the problem of orphan name servers. This resulted in a recent presentation at an ICANN meeting [82], and was based on data collected by Karmasphere [44] and Internet Identity [36]. Their work begins with lists of malicious domains and checks if any name servers used by them are orphans. They find 3.4% of phishing domains and 59% of fast flux domains use orphans. However, because they start with malicious domains, they do not identify orphans used for benign purposes. We take a different approach, starting with Top Level Domain (TLD) zone files instead of just with malicious domains, in order to find the general prevalence of orphans.

9.1.4 Malicious Autonomous Systems

Our work on identifying malicious ASes is motivated by recent disconnections of malicious ASes from the Internet due to malicious activities. These include Atrivo [31] and the web hosting provider McColo [32], both shut down in 2008 by their upstream providers, as well as Pricewert [10], shut down in 2009 by the Federal Trade Commission (FTC). In all three cases, the networks were accused of large amounts of botnet activity, malware hosting, and spamming. While these three attracted enough attention for high profile action against them and coverage by much of the technology news media, we aim to determine to what extent malicious activity is clustered together in other ASes which have not received as much attention or drastic action.

Other work touches on AS locations of malicious behaviors on the Internet. In a paper on spammers' behaviors, Ramachandran *et al.* [87] find that a small number of ASes are responsible for sending a large amount of spam, with 36% of all spam coming from just 20 ASes out of the approximately 30,000 total ASes on the Internet today. Konte *et al.* [47] examined scam hosting infrastructure. Among their findings was that for the spam campaigns they examined there was almost no overlap in the ASes of the spamming machines and the ASes where the scam web sites were hosted. These papers do not focus on the AS locations of the behavior in detail as we do. One other paper, by Stone-Gross *et al.* [100] does. They used a different method of measuring AS maliciousness than either of ours, and did not examine BGP behaviors or relationships among the ASes identified.

Some works attempt to locate malicious behavior at granularities other than ASes. In their study of spyware, Moshchuk *et al.* [65] find that certain categories of web sites, such as those offering free downloadable games, contain more spyware than others. Similarly, work by Provos *et al.* [84] finds that 67% of malware download sites in drive-by downloads are hosted in a single country; China. While there is insight to be gained by examination at these other granularities, locating malicious behavior by web site category or by country likely does not narrow the location down small enough to be useful for identification.

9.1.5 Web Redirects

Client-side defenses are necessary in cases when legitimate web sites fail to protect their redirects from manipulation of the redirect destination. Today, phishing toolbars can detect the final destination of the redirection chain and block access to known phishing sites [72]. However, the phishing site must be blacklisted before such toolbars can operate, which does not immediately protect users from deception. Likewise, a blacklist of open redirect web pages would have similar limitations. Redirect Remover [116], a Firefox browser extension, analyzes links on the page client is visiting and rewrites them to expose the actual destination. Unfortunately, this breaks some of the legitimates uses of redirects. Further, it does not protect against phishing, where the redirected links come from email messages.

Fette et al. [20] describe open redirects in email as an indication of phishing. They use this to motivate one of their heuristics for detecting phishing emails based on the number of dots contained in Uniform Resource Locators (URLs) in the email. Wang et al. [110] analyze web sites to detect malicious sites that exploit browser vulnerabilities. In doing so, they analyze whether web site redirects are being used to obfuscate the attack. They find that many sites hosting exploits hide behind redirects. When following redirects, their list of exploit providers grew 263% larger than the exploit providers they found scanning URLs without following redirects.

Netcraft provides a commercial service to check web sites for open redirects [72]. They additionally provide examples of previously found open redirects. However, they do not provide details of their methods or any information on what they find beyond the small set of motivating examples.

9.2 Malicious Infrastructure

Various characteristics of malicious web sites have been investigated in previous work. Some of these could likely be incorporated as features to detect malicious activity based on infrastructure.

Ramachandran and Feamster [87] examine the network level behaviors of spammers. They find that a few portions of IP address space send most spam, most hosts that send spam only send a few messages each, and spammers are taking advantage of short-lived route announcements. In this

last practice, spammers announce IP address space using Border Gateway Protocol (BGP), send the spam, and then withdraw the announcement. This practice can make the origin of the spam message untraceable. These short-lived announcements often involve address space not belonging to the spammer announcing it. This case is called BGP prefix hijacking, and it likely has no legitimate use. Several papers, including work by Zhang *et al.* [122] and many others, seek to identify hijacking attempts. Zhang *et al.* also have work on defending against hijacking attempts [121]. Several other systems, such as SBGP [45] and soBGP [112], also prevent prefix hijacking. Although we did not investigate it, prefix hijacking could indicate malicious behavior (although it could also happen due to configuration error), and may be a good infrastructure feature to use for our purposes.

Anderson *et al.* [4] attempt to characterize scam hosting infrastructure on the Internet. They apply an *image shingling* technique which allows them to determine the graphical similarity of web pages. They use this technique to find that 36,390 unique URLs in their spam feed lead to just 2,334 distinct scams. They also find that most scams from their trace are hosted on a single IP address and one domain name. Unfortunately, while their goal was to look for infrastructure characteristics, nothing they find can be easily used as a feature in our framework aside from the observation that most spam relays and scam hosts fall into two address ranges. However, the ranges are large and they do not discuss if those ranges are more populated in general than most. Other work, such as that by Lui *et al.* [111], and Medvet *et al.* [59] also propose schemes that use visual similarity. Both look for similarity of elements on the page, such as similar text colors, fonts, and alignment, and similar image sizes and color. They also look for overall page similarity. Although these approaches are more content-based than the ones we examine, they are also taking advantage of a behavior very beneficial to the miscreant: imitating a legitimate page. Because of the low sample sizes used in these papers for evaluation, the true effectiveness of these approaches can not be determined. If the comparison could be done quickly enough, this feature may be a useful addition for our framework.

McGrath and Gupta [57] examine URLs used in phishing attacks and the attack lifetimes. They find that 50-75% of phishing URLs contain brand names, that phishing domain names are shorter than good domains names, and that phishing URLs are generally longer than good URLs. They also find that letter frequencies in phishing URLs differ significantly from letter frequencies in English, while those in a corpus of good URLs are similar to that of English. These simple characteristics

would be easy to test for. Garera *et al.* [22] also examine phishing URLs. They find that the name of the organization being phished is often contained in the URL, but not often in the host name. When the name is contained in the host name, the host name is very long. They also look for the URLs in Google's database, based on the assumption that newer URLs will not be indexed, and phishing URLs will have a low Page Rank [79]. Whittaker *et al.* [113] examine some of these same features, plus others such as common strings contained in URLs aside from just the bank name and common terms appearing on the web page. They claim a false positive rate of below 0.1%, however, their system relies on a whitelist for a large portion of its classifications, and is only trained and evaluated against biased data, mainly URLs from emails marked by users as spam.

Zhang *et al.* [120] present a system called Cantina, which uses content to detect phishing web sites. While they are primarily detecting based on content, and not on Infrastructure, some of the features they look at are appropriate for our framework. One of these is the age of the domain. They find that phishing web sites are often registered only a few days before phishing emails are sent. They also check for the number of dots in a URL, similar to the length heuristic discussed above, for suspicious characters such as an @ in the URL, and for an IP address instead of a domain name. Another feature they look for is forms asking for personal information. These can all be considered infrastructure features which should be simple to check, and so can be used in our framework. Fette *et al.* [20] use a similar set of features as Cantina to detect phishing emails, including the age of domain names linked to, IP address based URLs, and links with a URL in the text not matching the actual destination of the link.

Other recent work takes a proactive approach to finding the domain names likely to be used by malicious web sites. Felegyhazi *et al.* [19] searches for domain names registered at the same time as malicious ones. They then check if these domains use the same DNS servers and change DNS servers at the same time as known malicious ones. If so, they infer that these are likely to belong to the same people and are therefore likely also malicious. While this is an interesting method of finding malicious domain names based on infrastructure, those it finds will be as old as some already blacklisted. While better than current blacklisting, it could still be identifying these domains too late. Sato *et al.* [91] use DNS in a different way to find domains to add to blacklists. They examine DNS queries, and consider a domain name to be likely malicious if it is queried by many of the

same hosts as a known malicious domain. Work by Prakash *et al.* [83] uses a different method of generating suspected new malicious URLs. They use heuristics based on similarities of existing blacklisted URLs to generate others. As with the previous system, it is likely that the URLs found this way may already be in use, but not yet blacklisted. In fact, they find that many of the generated URLs are in use with similar content.

Work on understanding botnets is important to us as it is believed that scams are often hosted on bots. Gu *et al.* [27] present BotSniffer, a system for detecting botnets by detecting their command and control traffic. Detecting which hosts are bots would be a great addition to our framework. Unfortunately, BotSniffer relies on monitoring network traffic at the edge of the network containing the bots in order to find the bots. In other work, Gu *et al.* [26] present BotHunter, which detects bots a different way. They monitor traffic for the stages of a botnet infection, looking for inbound scans followed by a download and outbound attacks. As with BotSniffer, this system detects bots only in the network it is deployed on. This requirement makes both systems impractical to use in our framework. BotMiner, by Gu *et al.* [25] detects bots a different way by looking at communications patterns and malicious activity patterns. It then clusters separately based on these two types of patterns and declares hosts in the same cluster by both methods to be part of the same botnet. This system has the same drawback as the previous two. Rajab *et al.* [86] use DNS cache snooping to find botnets, in addition to monitoring command and control. Unfortunately, the DNS cache snooping can only indicate which networks have botnets, not the actual machine. Additionally it only works if the DNS server used on the network for internal queries answers external ones as well. Unfortunately, we know of no infrastructure measurement that can directly tell us which IP addresses are hosting bots without monitoring their traffic.

9.3 Other Approaches

Other approaches to scam, phishing, and malware are complementary to our framework. Scams can be mitigated by detection of spam emails. Additionally, other methods of characterizing malicious URLs are possible. We discuss some of these complementary approaches here.

Spam is a key method used by miscreants to draw visitors to their sites. Various approaches have

been taken to identify unauthorized mail senders in order to prevent spam. Three of these which operate through DNS are Sender Policy Framework (SPF) [114], DomainKeys [14], and SenderID [51]. All of these basically publish records in DNS that allows a recipient to check if the sender was authorized to send mail for the domain the mail appears to be coming from. These methods rely on the recipient and the sender's claimed domain both participating in the scheme, while we try to rely on features that do not require participation from any outside entities. They also try to prevent the spam mail itself, while we are more focused on the web.

Ramachandran *et al.* [88] present a system called Spamtracker, which identifies spam based on sender behavior rather than the sender's IP address. They cluster spammer IP addresses based on the set of domains spam mails were sent to from those IP addresses. They then determine if a mail is spam by checking if the sender has sent mail to a set of domains similar to one of the clusters. This approach requires access to the email logs of a large number of domains, so it may not be practical for many organizations. It also relies on the assumption that spammers are consistent with the set of domains they target.

Xie *et al.* [117] present a system called AutoRE, which clusters spam in a different way. They look at the URLs contained in the spam messages, and cluster together messages that are similar. Further, they identify botnets sending spam by looking for the same URL signature in messages sent simultaneously from at least 20 ASes. They are able to identify 7,721 spam campaigns and 340,050 botnet IP addresses in this manner. However this system requires a large amount of email to work on.

Hao *et al.* [28] design a system called SNARE, with the goal of detecting spam based on network level features. They use several features to detect spam, some of which only require a single packet. Some of these are related to the features we use for detecting malicious web sites, such as how far geographically spam travels from the sending machine to the target, and what AS they are contained in. They claim a 70% detection rate of spam emails, with only a 0.3% false positive rate. However, they focus on spam, while we focus on the scam web sites it may lead to. Additionally, some of their features require aggregate data, such as mean and variance of message sizes from a given sender.

Email is not the only delivery mechanism for spam. Niu *et al.* [75] examine forum spam. They

find forum spamming to be widespread, with spam often posted on older pages for the purpose of manipulating search engines. They also find a large percentage of blogs on three blog hosting sites to exist for the sole purpose of spam.

Ma *et al.* [52, 53] present a system with similar goals to ours, proactive identification of malicious URLs, but a different method of reaching that goal. We aim to identify malicious web sites based on techniques beneficial to the miscreants. As such, our features should each apply to many malicious web sites. Because of this, we believe our framework is viable with only a small number of features, and the feature set should be relatively stable. The approach taken by Ma *et al.*instead relies on an extremely large number of features, mostly based on the URL itself. The biggest difference in their approach is the specificity of the features they use. For example, each individual domain name seen is a feature in their system, as is each IP prefix, each AS number, and each TLD. This leads to an extremely large number of feature, over half a million after running their system for ten days. Because of this, the model they use to classify new URLs needs to be updated frequently. In contrast, our set of features is stable, only changing when miscreants adopt new fundamentally different provisioning strategies. We believe our approach is also more robust to non-fundamental changes in operations. Their system may have difficulty if a group of miscreants were to change many minor characteristics such as their path structure, the hosts they are using, and their host names all at once. However, ours is likely to be more robust to these small changes, as long as the fundamentals activities continue such as use of fast flux to avoid take-down and use of many host names through DNS wildcards for ease in evading blacklists.

A large amount of work examines integrating anti-phishing tools into the web browser. Many of these simply rely on blacklists. Kirda and Krugel [46] take a different approach. Instead of identifying malicious domains, their plug-in, Anti-Phish, stores information entered in forms and alerts the user to a phishing attempt if the same information is entered in a form on a different site. Such an approach is likely to have high false positives in practice. Many browser toolbars to detect phishing exist, along with blacklist based solutions built in to Microsoft Internet Explorer [60] and Mozilla Firefox [66]. Ten of these are compared by Zhang *et al* [119]. They find that only two correctly identified more than 60% of phishing URLs, and one of these also flagged 42% of legitimate URLs as phishing. Chou *et al.* [11] created a browser plug-in called Spoofguard, notable because it

does not use blacklists or whitelists. Instead it determines if a page is a web spoofing page based on characteristics of the domain name, URL, and images. Spoofguard identified the most phishing pages in Zhang's work, but also had the highest false positives. McAfee site advisor [56] goes beyond trying to identify phishing as most of these tools do, and also examines sites for downloads of spyware, adware, other malware, and for if the sites it links to are considered good.

Moshchuk *et al.* [65] examine the prevalence of spyware on the web. They perform two web crawls. In the first, they crawled 18 million URLs and found that 13% of the 21,200 executables they found contained spyware. They also found drive-by downloads on almost 6% of pages they crawled. In their second crawl, they found a reduction in infected files and in drive by downloads, although they found more unique spyware programs. Although the reduction in infected files was substantial, down to 5.5%, they attribute the reduction to a single site being cleaned. They also examine which of the sites serving spyware are blacklisted, and find that blacklists concentrate on sites serving certain kinds of malware, and focus on the heavy hitters.

Network based intrusion detection systems (IDSes) use signatures capturing a wide variety of malicious behaviors. However, their goal is to identify malicious traffic instead of malicious sites, so they look at traffic properties for their characteristics. Snort [90, 95] signatures consist of values to look for in packet headers, as well as fixed strings to match in the packet payload. It allows plug-ins to extend its functionality. Bro [49, 81] uses regular expressions in its signatures, has its own language for specifying the signatures easily, and can incorporate analysis of previous activity into its analysis of current activity. Both aim to be easy to use, yet be able to use a large number of signatures to capture a large variety of attacks, and fast enough to analyze traffic at line speed.

10

Conclusion

As increasing numbers of people use the Internet for transactions such as shopping and banking, more and more opportunities arise for them to make a simple mistake and fall for some scam or give out personal identification to the wrong people. Safety against malicious activities on the Internet is an important concern and one which will only continue to grow.

In this dissertation, we proposed a framework for identifying malicious web sites by identifying infrastructure provisioning practices beneficial to miscreants. Such a framework would be an ideal complement to existing reactive techniques such as blacklisting. The reactive techniques can ensure that known attacks get blocked, while our technique will catch a large number of new attacks, ensuring they do not victimize users before the reactive techniques are adapted to detect them.

10.1 Contributions

We have identified five specific infrastructure provisioning practices, each of which is beneficial to miscreants and can be used to identify malicious web sites. A few of the practices we identify have legitimate uses in addition to the malicious ones. This is expected, since the Internet is a diverse system, and operators of various networks face different constraints. Although this presents a challenge, we have shown that it can be overcome by identifying additional characteristics that assist in their distinguishing the good and bad uses. For example, when examining the use of Domain

Name System (DNS) wildcards, the ratio of Autonomous Systems (ASes) and IP addresses pointed to by the wildcard records, along with their Time to Live (TTL) can be used for this purpose. Further, we believe that combining multiple features may also help, since sophisticated miscreants may use techniques such as fast flux to protect against blacklisting or removal of IP addresses, along with wildcards to protect against blacklisting of host names. Determining how often such situations occur, however, is left for future work.

Malicious activities on the Internet are ever-changing. One criticism against our work could be that malicious activity may evolve, making some of the features we investigated obsolete. For example, miscreants exploiting fast flux could use fewer hosts at a time or a better host selection algorithm to complicate its detection, or could abandon the practice altogether in favor of other strategies. Orphans may be stopped through policy decisions and enforcement by Top Level Domain (TLD) operators. We have two responses to such criticism. First, if we drive miscreants to abandon some of these practices, at least we have made their activities more difficult. Second, the actual features we identified are not as important for themselves as they are as a demonstration that such features exist. If miscreants move on to different activities, other features could be developed which identify those. We expect there will always be unique practices miscreants use to gain an advantage, and have shown that at least in the current Internet, there certainly are. We can take advantage of these, whatever they may be, to proactively protect users from a large portion of malicious activity.

10.2 Future Work

While we have shown that the framework we describe is promising, there are still several avenues for future work. We describe a few here.

10.2.1 Combinations of features

While we examined five specific provisioning practices, some of them are in use by good sites as well. We believe that this issue can be solved through the addition of more features, and an exploration of which features often occur in combination with each other. If multiple features often occur in

combination with each other, then these combinations could be used as an even stronger indication that a web site is malicious. We believe that sophisticated attackers will look for any advantage they can get to keep their activities alive as long as possible, so we expect that such combinations could be found.

10.2.2 Identifying hacked sites

The features we propose mostly assume that miscreants have provisioned their own infrastructure, although possibly on compromised user machines, or bots not actually belonging to them. However, other methods of setting up malicious web sites are also prevalent. Specifically, miscreants may hack into legitimate web sites and exploit them in their campaigns. In cases where miscreants provision their campaigns on the hacked sites, our framework may not be effective, since miscreants may not have access to the DNS infrastructure of the hacked site. One improvement that can be made based on this is to identify features of these hacked sites.

10.2.3 Prototype System

Perhaps our most ambitious plan for future work is building a prototype of the proposed framework. We will begin by training a classifier on known malicious web sites based on the infrastructure features we investigated, and testing it on other known blacklisted web sites. Assuming success, we will then move on to investigating other aspects of the proposed system. These include collecting new Uniform Resource Locators (URLs) quickly enough to mark them as malicious before they can do significant damage, and examining the overheads introduced to the DNS system by the extra queries an implementation of our framework would generate.

Bibliography

[1] Greg Aaron and Rod Rasmussen. APWG global phsihing survey: Trends and domain name use in 2h2009, May 2010. http://www.antiphishing.org/reports/APWG_GlobalPhishingSurvey_2H2009.pdf.

[2] Afilias Limited. How can I get access to Afilias' TLD zone file for .INFO domains? http://www.info.info/faq/how-can-i-get-access-afilias-tld-zone-file-info-domains.

[3] Amazon.com, Inc. Alexa web information service (AWIS). http://aws.amazon.com/awis.

[4] David Anderson, Chris Fleizach, Stefan Savage, and Geoffrey Voelker. Spamscatter: Characterizing Internet scam hosting infrastructure. In *USENIX Security Symposium*, 2007.

[5] APWG. Anti-phishing working group. http://www.antiphishing.org/.

[6] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. Resource records for the DNS security extensions. IETF RFC 4034, March 2005.

[7] John Bambenek. Double flux defense in the DNS protocol. IETF Internet Draft http://tools.ietf.org/html/draft-b ambenek-doubleflux-01, May 2008.

[8] Alper Caglayan, Mike Toothaker, Dan Drapeau, and Dustin Burke a nd Gerry Eaton. Real-time detection of fast flux service networks. In *Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, 2008.

[9] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[10] Jacqui Cheng. FTC forces hive of scum and villainy ISP offline, 2009. `http://arstechnica.com/tech-policy/news/2009/06/ftc-forces-hive-of-scum-and-villainy-isp-offline.ars`.

[11] Neil Chou, Robert Ledesma, Yuka Teraguchi, and John Mitchell. Client-side defense against web-based identity theft. In *Network and Distributed System Security Symposium (NDSS)*, 2004.

[12] Consumer Reports. Social insecurity: What millions of online users don't know can hurt them, June 2010. `http://www.consumerreports.org/cro/magazine-archive/2010/june/electronics-computers/social-insecurity/overview/index.htm`.

[13] Leslie Daigle. WHOIS protocol specification. IETF RFC 3912, September 2004.

[14] Mark Delany. Domain-based email authentication using public keys advertised in the DNS (DomainKeys). IETF RFC 4870, May 2007.

[15] DMOZ. Open directory project. `http://www.dmoz.org/`.

[16] DotAsia Organization Limited. .ASIA Zone File Access Agreement. `http://www.dotasia.org/info/DAO.ZONE-2007-10-24.pdf`.

[17] Robert Elz, Randy Bush, Scott Bradner, and Michael Patton. Selection and operation of secondary DNS servers. IETF RFC 2182, July 1997.

[18] eSoft Inc. `http://www.esoft.com/`.

[19] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[20] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *International World Wide Web Conference (WWW)*, 2007.

[21] Lixin Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions Of Networking*, 9(6):733–745, December 2001.

[22] Sujita Garera, Niels Provos, Monica Chew, and Aviel Rubin. A framework for detection and measurement of phishing attacks. In *ACM Workshop on Recurring Malcode (WORM)*, 2007.

[23] Google. Google AJAX Search API. `http://code.google.com/apis/ajaxsearch/`.

[24] Google Code Labs. Google safe browsing API. `http://code.google.com/apis/safebrowsing/`.

[25] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. BotMiner: Clustering analysis of network traffic for protocol and structure-independent botnet detection. In *USENIX Security Symposium*, 2008.

[26] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. BotHunter: Detecting malware infection through ids-drived dialog correlation. In *USENIX Security Symposium*, 2007.

[27] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting command and control channels in network traffic. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[28] Shuang Hao, Nadeem Syed, Nick Feamster, Alexander Gray, and Sven Krasser. Detecting spammers with SNARE: Spatio-temporal network-level automated reputation engine. In *USENIX Security Symposium*, 2009.

[29] Hexasoft Development Sdn. Bhd. IP2Location geolocation service. http://www.ip2location.com/, February 2008.

[30] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix Freiling. Measuring and detecting fast-flux service networks. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[31] Joel Hruska. Bad seed ISP Atrivo cut off from rest of the Internet, 2008. `http://arstechnica.com/security/news/2008/09/bad-seed-isp-atrivo-cut-off-from-rest-of-the-internet.ars`.

[32] Joel Hruska. Spam sees big nosedive as rogue ISP McColo knocked offline, 2008. http://arstechnica.com/security/news/2008/11/spam-sees-big-nosedive-as-rogue-isp-mccolo-knocked-offline.ars.

[33] Internet Architecture Board. Architectural concerns on the use of DNS wildcards. IAB Commentary, September 2003. www.iab.org/documents/docs/2003-09-20-dns-wildcards.html.

[34] Internet Corporation for Assigned Names and Numbers (ICANN). Update: ICANN projects underway to improve whois accuracy. http://www.icann.org/announcements/announcement-2-21dec07.htm.

[35] Internet Crime Complaint Center. 2009 Internet crime report, 2010. http://www.ic3.gov/media/annualreport/2009_IC3Report.pdf.

[36] Internet Identity. http://www.internetidentity.com.

[37] Internet Systems Consortium, Inc. ("ISC"). *BIND 9 Administrator Reference Manual (9.3.2)*, 2005. http://www.bind9.net/manuals.

[38] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series, 2nd edition, 1988.

[39] Simon Josefsson. DNSSEC walker. http://josefsson.org/walker/.

[40] Andrew Kalafut, Minaxi Gupta, Christopher Cole, Lei Chen, and Nathan Myers. An empirical study of orphan DNS servers in the Internet. Under submission.

[41] Andrew Kalafut, Minaxi Gupta, Pragneshkumar Patel, Pairoj Rattadilok, Yasir Ibrahim, and Cuifang Lin. Surveying DNS wildcard usage among the good, the bad, and the ugly. In *ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, 2010.

[42] Andrew Kalafut, Craig Shue, and Minaxi Gupta. Understanding implications of DNS zone provisioning. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, 2008.

[43] Andrew Kalafut, Craig Shue, and Minaxi Gupta. Malicious hubs: Detecting abnormally malicious autonomous systems. In *IEEE Conference on Computer Communications (INFOCOM) Mini-Conference*, 2010.

[44] Karmasphere, Inc. `http://karmasphere.com/`.

[45] Stephen Kent, Charles Lynn, and Karen Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, 2000.

[46] Engin Kirda and Christopher Kruegel. Protecting users against phishing attacks. *The Computer Journal*, 49(5):544–561, January 2006.

[47] Maria Konte, Nick Feamster, and Jayeon Jung. Dynamics of online scam hosting infrastructure. In *Passive and Active Measurement Conference*, 2009.

[48] Brian Krebs. Major source of online scams and spams knocked offline, 2008. `http://voices.washingtonpost.com/securityfix/2008/11/major_source_of_online_scams_a.html`.

[49] Lawrence Berkeley National Laboratory. Bro intrusion detection system. `http://bro-ids.org/`.

[50] Edward Lewis. The role of wildcards in the domain name system. IETF RFC 4592, July 2006.

[51] Jim Lyon and Meng Weng Wong. Sender ID: Authenticating e-mail. IETF RFC 4406, April 2006.

[52] Justin Ma, Lawrence Saul, Stefan Savage, and Geoffrey Voelker. Beyond blacklist: Learning to detect malicious web sites from suspicious URLs. In *ACM SIGKDD Conference on Knowldege Discovery and Data Mining*, 2009.

[53] Justin Ma, Lawrence Saul, Stefan Savage, and Geoffrey Voelker. Identifying suspicious URLs: An application of large-scale online learning. In *International Conference on Machine Learning*, 2009.

[54] Malware Patrol. Malware block list. `http://www.malwarepatrol.net/lists.shtml`.

[55] MarkMonitor, Inc. `http://www.markmonitor.com`.

[56] McAfee. McAfee SiteAdvisor. `http://www.siteadvisor.com/`.

[57] D. Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of the phisher modi operandi. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[58] D. Kevin McGrath, Andrew Kalafut, and Minaxi Gupta. Phishing infrastructure fluxes all the way. *IEEE Security and Privacy Magazine Special Issue on Securing DNS*, September/October 2009.

[59] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual similarity-based phishing detection. In *ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, 2008.

[60] Microsoft. Phishing filter: Help protect yourself from online scams. `http://www.microsoft.com/protect/products/yourself/phishingfilter.mspx`.

[61] Microsoft. Windows Live Fact Sheet. `http://www.microsoft.com/presspass/newsroom/msn/factsheet/WindowsLive.mspx`.

[62] Rich Miller. Phishers use wildcard DNS to build convincing bait URLs, March 2005. `http://news.netcraft.com/archives/2005/03/07/phishers_use_wildcard_dns_to_build_convincing_bait_urls.html`.

[63] Paul Mockapetris. Domain names - concepts and facilities. IETF RFC 1034, November 1987.

[64] Paul Mockapetris. Domain names - implementation and specification. IETF RFC 1035, November 1987.

[65] Alexander Moushchuk, Tanya Bragin, Steven Gribble, and Henry Levy. A crawler-based study of spyware on the web. In *Network and Distributed System Security Symposium (NDSS)*, 2006.

[66] Mozilla. Firefox phishing and malware protection. `http://www.mozilla.com/en-US/firefox/phishing-protection/`.

[67] Mozilla Foundation. Public suffix list. `http://publicsuffix.org`.

[68] mTLD, Ltd. dotMobi Zone File Access Agreement. `http://mtld.mobi/domain/zonefile`.

[69] Paul Mutton. New phishing attacks combine wildcard DNS and XSS, February 2009. `http://news.netcraft.com/archives/2009/02/17/new_phishing_attacks_combine_wildcard_dns_and_xss.html`.

[70] John Nagle. Metrics of collateral damage from blacklisting of domains exploited by "phishing" operations. In *MIT Spam Conference*, 2008.

[71] Jose Nazario and Thorsten Holz. As the net churns: Fast-flux botnet observations. In *International Conference on Malicious and Unwanted Software (MALWARE)*, 2008.

[72] Netcraft. Netcraft: Anti-fraud open redirect detection service. `http://news.netcraft.com/open-redirect-detection`.

[73] NETpilot GmbH. Viruswatch mailing list. `http://lists.clean-mx.com/cgi-bin/mailman/listinfo/viruswatch`.

[74] NeuStar Registry Services. .BIZ Zone File Distribution. `https://www.neulevel.biz/zonefile/`.

[75] Yuan Niu, Yi-Min Wang, Hao Chen, Ming Ma, and Francis Hsu. A quantitative study of forum spamming using content-based analysis. In *Network and Distributed System Security Symposium (NDSS)*, 2007.

[76] University of Oregon Advanced Network Technology Center. Route Views project. `http://www.routeviews.org/`.

[77] OpenDNS. PhishTank. `http://www.phishtank.com/`.

[78] Eric Osterweil, Dan Massey, and Lixia Zhang. SecSpider. `http://secspider.cs.ucla.edu`.

[79] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[80] Emanuele Passerini, Roberto Paleari, Lorenzo Martignoni, and Danilo Bruschi. Fluxor: detecting and monitoring fast-flux service networks. In *SIG SIDAR Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2008.

[81] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23):2435–2463, December 1999.

[82] Dave Piscitello. Orphaned name servers. ICANN 35 Presentation, June 2009.

[83] Pawan Prakash, Manish Kumar, Ramana Kompella, and Minaxi Gupta. PhishNet: Predictive blacklisting to detect phishing attacks. In *IEEE Conference on Computer Communications (INFOCOM) Mini-Conference*, 2010.

[84] Niels Provos, Panayiotis Mavrommatis, Moheeb Rajab, and Fabian Monrose. All your iFRAMEs point to us. In *USENIX Security Symposium*, 2008.

[85] Public Interest Registry. .ORG Registry - Zone File Access. `http://pir.org/index.php?db=content/Website&tbl=Registrars&id=7`.

[86] Moheeb Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, 2006.

[87] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *ACM SIGCOMM*, 2006.

[88] Anirudh Ramachandran, Nick Feamster, and Santosh Vemplana. Filtering spam with behavioral blacklisting. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[89] Rod Rasmussen and Greg Aaron. APWG global phsihing survey: Trends and domain name use in 1h2009, October 2009. `http://www.antiphishing.org/reports/APWG_GlobalPhishingSurvey_1H2009.pdf`.

[90] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Large Installation System Administration Conference (LISA)*, 1999.

[91] Kazumichi Sato, Keisuke Ishibashi, Tsuyoshi Toyono, and Nobukisa Miyake. Extending black domain name list by using co-occurrence relation between DNS queries. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[92] ShadowServer Foundation. `http://www.shadowserver.org/wiki/`.

[93] C. Shue, A. Kalafut, and M. Gupta. Exploitable redirects on the web: Identification, prevalence, and defense. In *USENIX Workshop on Offensive Technologies*, 2008.

[94] Sophos, Plc. Email security products. `http://www.sophos.com/products/enterprise/email/`.

[95] Sourcefire. Snort - the de facto standard for intrusion detection/prevention. `http://www.snort.org/`.

[96] Spamhaus Project. Exploits block list (XBL). `http://www.spamhaus.org/xbl/index.lasso`.

[97] Spamhaus Project. Register of known spam operations - canadian pharmacy. `http://www.spamhaus.org/rokso/evidence.lasso?rokso_id=ROK8138`.

[98] Spamhaus Project. Spamhaus block list (SBL). `http://www.spamhaus.org/sbl/index.lasso`.

[99] SpamSuite.com. Findings of fact, conclusions of law, and order for judgment. `http://www.spamsuite.com/node/351`.

[100] Brett Stone-Gross, Christopher Kruegel, Kevin Almeroth, Andreas Moser, and Engin Kirda. FIRE: Finding rogue networks. In *Annual Computer Security Applications Conference (ACSAC)*, 2009.

[101] Support Intelligence, LLC. `http://www.support-intelligence.com/`.

[102] SURBL. `http://www.surbl.org/`.

[103] Team Cymru, Inc. IP to ASN mapping. `http://www.team-cymru.org/Services/ip-to-asn.html`.

[104] The Honeynet Project. Know your enemy: Fast-flux service networks. Whitepaper, July 2007. `http://www.honeynet.org/papers/ff`.

[105] VeriSign. Domain name industry brief, June 2007. `http://www.verisign.com/static/042161.pdf`.

[106] VeriSign. Domain name industry brief, February 2009. `http://www.verisign.com/static/044518.pdf`.

[107] VeriSign. Domain name industry brief, June 2010. `http://www.verisign.com/domain-name-services/domain-information-center/domain-name-resources/domain-name-report-june10.pdf`.

[108] VeriSign. Domain name industry brief, February 2010. `http://www.verisign.com/domain-name-services/domain-information-center/domain-name-resources/domain-name-report-feb10.pdf`.

[109] VeriSign, Inc. TLD Zone Access Program. `http://www.verisign.com/domain-name-services/domain-information-center/tld-zone-access/`.

[110] Yi-Min. Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King. Automated web patrol with Strider HoneyMonkeys. In *Network and Distributed System Security Symposium (NDSS)*, 2006.

[111] Liu Wenyin, Guanglin Huang, Liu Ziaoyue, Zhang Min, and Xiaotie Deng. Detection of phishing webpages based on visual similarity. In *International World Wide Web Conference (WWW)*, 2005.

[112] Russ White. Securing BGP through secure origin BGP (soBGP). *The Internet Protocol Journal*, 6(3):15–22, 2003.

[113] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *Network and Distributed System Security Symposium (NDSS)*, 2010.

[114] Meng Weng Wong and Wayne Schlitt. Sender policy framework (SPF) for authorizing use of domains in e-mail, version 1. IETF RFC 4408, April 2006.

[115] World Wide Web Consortium (W3C). Use standard redirects - don't break the back button! `http://www.w3.org/QA/Tips/reback`.

[116] Xeen. Redirect remover. `http://redirectremover.mozdev.org/`.

[117] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrathy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: Signatures and characteristics. In *ACM SIGCOMM*, 2008.

[118] Bojan Zdrnja, Nevil Brownlee, and Duane Wessels. Passive monitoring of DNS anomalies. In *SIG SIDAR Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2007.

[119] Yue Zhang, Serge Egelman, Lorrie Cranor, and Jason Hong. Phinding phish: Evaluating anit-phishing tools. In *Network and Distributed System Security Symposium (NDSS)*, 2007.

[120] Yue Zhang, Jason Hong, and Lorrie Cranor. CANTINA: a content based approach to detecting phishing web sites. In *International World Wide Web Conference (WWW)*, 2007.

[121] Zheng Zhang, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. Practical defenses against BGP prefix hijacking. In *Conference on Future Networking Technologies (CoNext)*, 2007.

[122] Zheng Zhang, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. iSPY: detecting IP prefix hijacking on my own. In *ACM SIGCOMM*, 2008.

Vita

Andrew J. Kalafut earned his B.S. in Computer Science and Mathematics from Bradley University in 2004. He then came to Indiana University where he received an M.S. in Computer Science in 2006. Andrew's research at Indiana University has focused on Internet security, particularly from the perspective of how miscreants provision malicious web sites. He has published ten papers in prominent security and networking journals and conferences, and his work has been covered by popular press venues including MIT Technology Review and the Washington Post. His research experience outside the University includes a summer internship at AT&T Labs Research. He has mentored multiple undergraduate and M.S. students, and was active in the Computer Science Graduate Student Association (CSGSA) and the Graduate and Professional Student Organization (GPSO).